CHƯƠNG 1

TỔNG QUAN VỀ MẠNG VÀ CÁC DỊCH VỤ THÔNG DỤNGTRÊN INTERNET

I. Động lực thúc đẩy sự ra đời mô hình mạng Client/Server :

- Ngày nay với xu hướng mạng toàn cầu hoá, thì sự liên lạc thông tin qua lại giữa các máy theo mô hình Client/Server là một trong những ứng dụng quan trong cơ bản về mạng và nó không thể thiếu trong hệ thống liên lạc thông tin hiện nay. Có rất nhiều dịch vụ hỗ trợ trên Internet theo mô hình này như e-mail, web, FPT, nhóm tin Usernet, telnet, truyền tập tin, đăng nhập từ xa, chat,… Các chương trình dịch vụ ở trình khách(Client) sẽ kết nối với trình chủ ở xa(Server) sau đó gởi các yêu cầu đến trình chủ và trình chủ sẽ xử lý yêu cầu này sau đó gởi kết quả về cho trình khách. Thông thường trình chủ phục vụ cho rất nhiều trình khách đến cùng một lúc.

- Vào những thập niên 90, khi bắt đầu bùng nổ sự truy cập Web cũng như mạng hoá trong các lĩnh vực của nhiều quốc gia trên thế giới trong đó có Việt Nam chúng ta. Một vấn đề đặt ra cho các nhà lập trình, các nhà quản lý và nhiều hơn nữa là những người sử dụng máy tính điều có thể truy cập thông tin trên Intranet hay Internet nhanh chóng, chính xác mà các thông tin hay dữ liệu này vẫn được an toàn. Lập trình mạng theo mô hình Client/Server sẽ là giải pháp an toàn cho các nhà lập trình.

II. Nguyên tắc hoạt động mạng theo mô hình client/Server :

- Mạng Client/Server đơn thuần chỉ có một tiêu chuẩn cơ bản là không có một Client nào sử dụng tài nguyên của một Client khác. Tài nguyên dùng chung (tài nguyên chính) được đặt trên một hay nhiều Server chuyên dụng theo từng dịch vụ như E-mail, file server, chat, Web, fpt,…hay nói một cách khác những Client không bao giờ nhìn thấy nhau mà chỉ giao tiếp với Server. Mô hình Client/Server này rất hữu dụng trong các công ty hay những tổ chức cần đến việc quản lý tài nguyên hay người sử dụng một cách hiểu quả.

- Thuật ngữ Server dùng để chỉ bất kỳ chương trình nào hỗ trợ dịch vụ có thể truy xuất qua mạng. Một Server nhận yêu cầu qua mạng thực hiện cho một dịch vụ nào đó và trả kết quả về cho nơi yêu cầu. Với những dịch vụ đơn giản nhất, mỗi yêu cầu gửi đến chỉ trong một địa chỉ IP datagram và Server trả về lời đáp trong một datagram khác. Các Server có thể thực hiện những công việc đơn giản nhất đến phức tạp nhất. Ví dụ như *time-of-day* Server chỉ đơn giản trả về giờ hiện hành bất cứ khi nào Client gởi tới Server này thông tin. Hay một Web Server nhận yêu cầu từ một trình duyệt (Borwser) để lấy một bản sao của trang web, Server sẽ lấy bản sao của tập tin trang web này trả về cho trình duyệt.

- Mô hình Client/Server thực hiện việc phân tán xử lý giữa các máy tính. Về bản chất là một công nghệ được chia ra và xử lý bởi nhiều máy tính, các máy tính được xem là Server thường được dùng để lưu trữ tài nguyên để nhiều nơi truy xuất vào. Các Server sẽ thụ động chờ để giải quyết các yêu cầu từ Client truy xuất đến chúng. Thông thường, các Server được cài đặt như một chương trình ứng dụng. Vì vậy ưu điểm của việc cài đặt các Server như những chương trình ứng dụng là chúng có thể xử lý trên hệ máy tính bất kỳ nào hỗ trợ thông tin liên lạc theo giao thức TCP/IP hay một giao thức thông dụng khác. Như thế, Server cho một dịch vụ cụ thể có thể chạy trên một hệ chia thời gian cùng với nhưng chương trình khác, hay nó có thể xử lý trên cả máy tính cá nhân.

- Một chương trình ứng dụng trở thành Client khi nó gởi yêu cầu tới Server và đợi lời giải đáp trả về. Cũng vì thế mà mô hình Client/Server là sự mở rộng

tự nhiên của tiến trình thông tin liên lạc trong nội bộ máy tính và xa hơn nữa là Intarnet/Internet. Ứng dụng đầu tiên của mô hình Client/Server là ứng dụng chia sẻ file(do các tổ chức có nhu cầu chia sẻ thông tin giữa các bộ phận trong tổ chức được dễ dàng và nhanh chóng hơn). Trong ứng dụng này thông tin được chứa trong các file đặt tại máy Server của một phòng ban nào đó. Khi một phòng ban khác có nhu cầu trao đổ thông tin với phòng ban này thì sẽ sử dụng một máy tính khác(Client) kết nối với Server và tải nhưng file cần thiết về máy Client.

Tóm lại :

+ Nhiệm vụ của máy Client : là thi hành một dịch vụ cho người dùng, bằng cách kết nối với những chương trình ứng dụng ở máy Server, dựa vào những chuỗi nhập để chuyển yêu cầu đến Server và nhân kết quả trả về từ Server hiển thị thông tin nhân được cho người dùng.

+ Nhiệm vụ của máy Server : luôn lắng nghe những kết nối đến nó trên những cổng liên quan đến giao thức mà Server phục vụ. Khi máy Client khởi tạo kết nối, máy Server chấp nhận và tạo ra luồng riêng biệt phục vụ cho máy Client đó. Ngoài ra máy Server phải quản lý các hoạt động của mạng như phân chia tài nguyên chung(hay còn gọi là tài nguyên mạng) trong việc trao đổi thông tin giữa các Client,… Máy Server có thể đóng vai trò là máy trạm (Client) trong trường hợp này gọi là máy Server *"không thuần tuý"*. Server phải đảm bảo được hai yêu cầu cơ bản nhất đối với chức năng Server : cho phép truyền dữ liệu nhanh chóng và bảo đảm tính an toàn, bảo mật và không mất mát dữ liệu.

+ Có thể nói mô hình Client/Server là mô hình ảnh hưởng lớn nhất tới ngành công nghệ thông tin. Mô hình này đã biến những máy tính riêng lẻ có khả năng xử lý thấp thành một mạng máy chủ(Server) và máy trạm(Workstation) có khả năng xử lý gấp hàng ngàn lần những máy tính mạnh nhất. Mô hình này còn giúp cho việc giải quyết những bài toán phức tạp một cách dễ dàng hơn, bằng cách phân chia bài toán lớn thành nhiều bài toán con và giải quyết từng bài toán con một. Nhưng quan trọng hơn hết, không phải là việc giải được các bài toán lớn mà là cách thức giải bài toán.

+ Ưu điểm:
  - Các tài nguyên được quản lý tập trung.
  - Có thể tạo ra các kiểm soát chặt chẽ trong truy cập file dữ liệu.
  - Giảm nhẹ gánh nặng quản lý trên máy Client.
  - Bảo mật và back up dữ liệu từ Server.

+ Nhược điểm:
  - Khá đắt tiền so với mạng ngang hàng(peer), chủ yếu do giá để lắp đặt một Server khá cao.
  - Server trở thành điểm tối yếu của hệ thống, nghĩa là khi Server hỏng thì toàn bộ hệ thống sẽ chết, do đó tính năng đề kháng lỗi là một trong những yêu cầu quan trọng trong mô hình này.

III.Các khái niệm cơ bạn về mạng :

- Ngày nay, chúng ta đã quá quen thuộc về mạng điện thoại trong việc trao đổi thông tin, tương tự mạng trong máy tính cũng sử dụng một số nguyên tắc cơ bản sau.

+ Bảo đảm thông tin không bị mất hay thất lạc trên đường truyền.

+ Thông tin được truyền nhanh chóng và kịp thời.

+ Các máy tính trong cùng một mạng phải nhận biết nhau.

+Cách đặt tên trên mạng cũng như cách xác định các đường truyền trên mạng phải tuân theo một chuẩn thống nhất.

- Các nguyên tắc trên có vẽ rất cơ bản nhưng nó hết sức quan trọng. Nhưng tại sao cần phải nối mạng? có nhiều lý do nhưng có thể kể các lý do sau:

+ Tăng hiệu quả làm việc.

+ Xây dựng mô hình làm việc thống nhất tập trung cho tất cả mọi người sử dụng mạng.

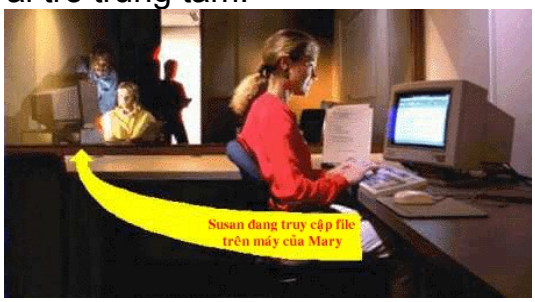
+ Cho phép đưa tất cả các vấn đề cần giải quyết lên mạng dưới dạng thảo luận theo quan điểm phóng khoáng, thoải mái hơn là phải đối thoại nhau trong một không khí gò bó.

+ loại bỏ các thông tin thừa, trùng lặp.

- Mạng có thể đơn giản chỉ gồm hai máy tính bằng cáp qua cổng máy in để truyền file, phức tạp hơn thì hiện nay có thể chia mạng ra thành các loại sau:

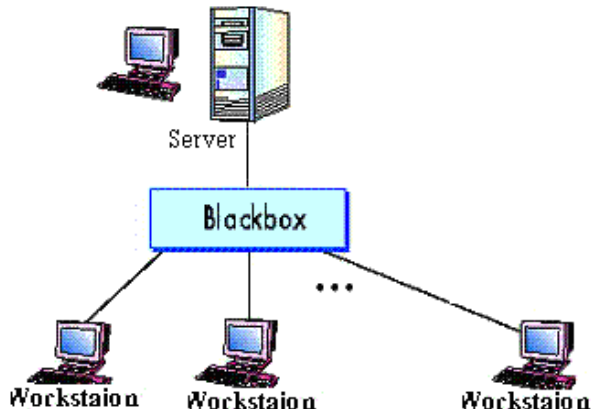
+ *Mạng cục bộ(LAN-Wide Area Network) :* là mạng đơn giản nhất trong thế giới mạng, là một hệ thống bao gồm các nút là các máy tính nối kết với nhau bằng dây cáp qua card giao tiếp mạng trong phạm vi nhỏ tại một vị trí nhất định. Tuỳ theo cách giao tiếp giữa các nút mạng, người ta chia làm hai loại :

- ***Mạng ngang hàng (peer to peer [Windows workgroups]) :*** là một hệ thống mà mọi nút đều có thể sử dụng tài nguyên của các nút khác. Nghĩa là các máy tính trên mạng đều ngang nhau về vai trò, không có máy nào đóng vai trò trung tâm.



Hình 1.1 : Marry đang truy xuất tài nguyên qua mạng

- ***Mạng khách chủ (client/server) :*** có ít nhất một nút trong mạng đảm nhiệm vai trò trạm dịch vụ (server) và các máy khác là trạm làm việc (workstation) sử dụng tài nguyên của các trạm dịch vụ. Server chứa hầu hết tài nguyên quan trọng của mạng và phân phối tài nguyên này tới các Client.



Hình 1.2 : Mô hình mạng Client/Server.

+ *Mạng đô thị(Metropolitan Area Networks - viết tắt là Man):* Là mạng đặt trong phạm vi một đô thị hoặc một trung tâm kinh tế-xã hội có bán kính khoản 100km trở lại. Là mạng chỉ với một đường truyền thuê bao tốc độ cao qua mạng điện thoại hoặc thông qua các phương tiện khác như radio, microway, hay các thiết bị truyền dữ liệu bàng laser. MAN cho phép người dùng mạng trên nhiều vị trí địa lý khác nhau vẫn có thể truy cập các tài nguyên mạng theo cách thông thường như ngay trên mạng LAN. Tuy nhiên nhìn trên phương diện tổng thể MAN cũng chỉ là mạng cục bộ.

+ *Mạng diện rộng(WAN – Wide Area Networks):* phạm vi của mạng vượt qua biên giới quốc gia và thậm chí cả lục địa. WAN có nhiệm vụ kết nối tất cả các mạng LAN và MAN ở xa nhau thành một mạng duy nhất có đường truyền tốc độ cao. Tốc độ truy cập tài nguyên của mạng WAN thường bị hạn chế bởi dung lượng truyền của đường điện thoại thuê bao(phần lớn các tuyến điện thoại số cũng chỉ ở mức 56 kilobits/s) và chi phí thuê bao rất đắt đây là vấn đề để cho một công ty hay tổ chức nào muốn thiết lập mạng MAN cho công ty mình.

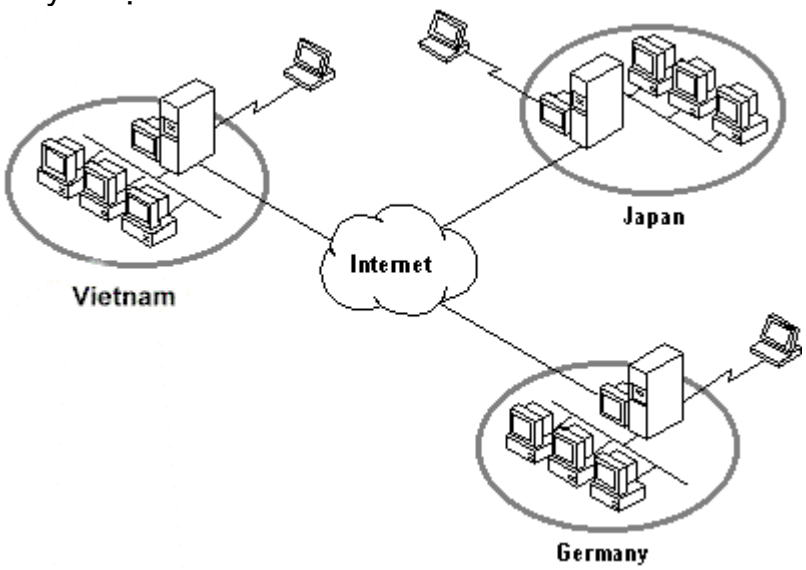
+ *Mạng Internet :*

- Mạng Internet là một tập hợp gồm hàng vạn mạng (LAN, MAN và WAN)trên khắp thế giới kết nối với qua một router(là thiết bị phân tuyến các luồn dữ liệu giữa các mạng) tạo thành một mạng chung trên toàn cầu theo mô hình client/Server, được phát triển vào đầu thập niên 70. Internet là công nghệ thông tin liên lạc mới, và hiện đại, nó tác động sâu sắc vào xã hội cuộc sống chúng ta, là một phương tiện cần thiết như điện thoại hay tivi, nhưng ở mức độ bao quát hơn. Chẳng hạn điện thoại chỉ cho phép trao đổi thông tin qua âm thanh, giọng nói. Với Tivi, thông tin nhận được trực quan hơn. Còn Internet đưa chúng ta vào thế giới có tầm nhìn rộng hơn và bạn có thể làm mọi thứ: viết thư, đọc báo, xem bản tin, giải trí, tra cứu và thậm chí còn thực hiện những phi vụ làm ăn, ….Vì Internet là mạng của các mạng, tức bao gồm nhiều mạng máy tính kết nối lại với nhau, Số lượng máy tính nối mạng và số lượng người truy cập vào mạng Internet trên toàn thế giới đang ngày càng tăng lên nhanh chóng. Đặc biệt từ năm 1993 trở đi, mạng Internet không chỉ cho phép chuyển tải thông tin nhanh chóng mà còn giúp cung cấp thông tin, nó cũng là diễn đàn và là thư viện toàn cầu đầu tiên. Các thông tin được đặt rải rác trên

toàn cầu có thể truyền thông được với nhau như một thiết bị Modem và đường dây điện thoại.

- Internet bắt đầu từ đầu năm 1969 dưới cái tên là ARPANET(Advanced Research Projects Agency) còn gọi là ARPA. Nó thuộc bộ quốc phòng Mỹ (DoD). Đầu tiên nó chỉ có 4 máy được thiết kế để minh hoạ khả năng xây dựng mạng bằng Cách dùng máy tính nằm rải rác trong một vùng rộng. Vào năm 1972, khi ARPANET được trình bày công khai, đã được 50 trường đại học và các viện nghiên cứu nối kết vào. Mục tiêu của ARPANET là nghiên cứu hệ thống máy tính cho các mục đích quân sự. Chính phủ và quân đội tìm kiếm những phương cách để làm cho mạng tránh được các lỗi, mạng này thiết kế chỉ cho phép các văn thư lưu hành từ máy tính này đến máy tính khác, đối với chính phủ và quân đội, máy tính đã có những công dụng rõ ràng và sâu rộng. Tuy nhiên, một trong những mối bận tâm chính yếu là tính đáng tin cậy vì nó có liên quan đến vấn đề sinh tử. Kế hoạch ARPANET đã đưa ra nhiều đường nối giữa các máy tính. Điều quan trọng nhất là các máy tính bạn có thể gởi các văn thư bởi bất kỳ con đường khả dụng nào, thay vì chỉ qua một con đường cố định. Đây chính là nơi mà vấn đề về giao thức đã xuất hiện. Đầu năm 1980 trung tâm DARPA thử nghiệm giao thức TCP/IP và được các trường đại học mỹ cho phép nối với hệ điều hành UNIX BSD (Berkely Software Distribution).
- Hệ điều hành UNIX là hệ phát triển mạnh với rất nhiều công cụ hỗ trợ và đảm bảo các phần mềm ứng dụng có thể chuyển qua lại trên các họ máy khác nhau (máy mini, máy tính lớn và hiện nay là máy vi tính). Bên cạnh đã hệ điều hành UNIX BSD còn cung cấp nhiều thủ tục Internet cơ bản, đưa ra khái niệm Socket và cho phép chương trình ứng dụng thâm nhập vào Internet một cách dễ dàng.
- Internet có thể tạm hiểu là liên mạng gồm các máy tính nối với nhau theo một nghi thức và một số thủ tục chung gọi là TCP/IP (Transmission Control Protocol/Internet Protocol).Thủ tục và nghi thức này trước kia đã được thiết lập và phát triển là cho một đề án nghiên cứu của Bộ Quốc Phòng Mỹ với mục đích liên lạc giữa các máy tính nối đơn lẻ và các mạng máy tính với nhau mà không phụ thuộc vào các hãng cung cấp máy tính. Sự liên lạc này vẫn được bảo đảm liên tục ngay cả trong trường hợp có nút trong mạng không hoạt động.
- Ngày nay, Internet là một mạng máy tính có phạm vi toàn cầu bao gồm nhiều mạng nhỏ cũng như các máy tính riêng lẻ được kết nối với nhau để có thể liên lạc và trao đổi thông tin. Trên quan điểm Client / Server thì có thể xem Internet như là mạng của các mạng của các Server, có thể truy xuất bởi hàng triệu Client. Việc chuyển và nhận thông tin trên Internet được thực hiện bằng nghi thức TCP/IP. Nghi thức này gồm hai thành phần là Internet protocol (IP) và transmission control protocol (TCP) (được nguyên cứu ở những phần sau). IP cắt nhỏ và đóng gói thông tin chuyển qua mạng, khi đến máy nhận, thì thông tin đó sẽ được ráp nối lại. TCP bảo đảm cho sự chính xác của thông tin được chuyền đi cũng như của thông tin được ráp nối lại đồng thời TCP cũng sẽ yêu cầu truyền lại tin thất lạc hay hư hỏng. Tuỳ theo thông tin lưu trữ và mục đích phục vụ mà các server trên Internet sẽ được phân chia thành các loại khác nhau như Web Server, email Server hay FTP Server. Mỗi loại server sẽ được tối ưu hoá theo mục đích sử dụng.

• Từ quan điểm người sử dụng, Internet trông như là bao gồm một tập hợp các chương trình ứng dụng sử dụng những cơ sở hạ tầng của mạng để truyền tải những công việc thông tin liên lạc. Chúng ta dùng thuật ngữ "interoperability" để chỉ khả năng những hệ máy tính nhiều chủng loại hợp tác lại với nhau để giải quyết vấn đề. Hầu hết người sử dụng truy cập Internet thực hiện công việc đơn giản là chạy các chương trình ứng dụng trên một máy tính nào đó gọi là máy client mà không cần hiểu loại máy tính(Server) đang được truy xuất, kỹ thuật TCP/IP, cấu trúc hạ tầng mạng hay Internet ngay cả con đường truyền dữ liệu đi qua để đến được đích của nó.Chỉ có những người lập trình mạng cần xem TCP/IP như là một mạng và cần hiểu một vài chi tiết kỹ thuật.



Hình 1.4 : Liên lạc trên Internet

Các kiểu kết nối Internet:

+ Kết nối quay số(dial-up connection): rẻ tiền nhất nhưng tốc độ truy cập bị hạn chế và có thể bị gián đoạn bất ngờ khi quá tải kênh truyền.

+ Kết nối qua các tuyến điện thoại có tốc độ truyền 56kbs/s tốc độ có khá hơn kiểu quay số nhưng không đáng kể.

Tuy nhiên, với tốc độ phát triển cực kỳ nhanh chóng của nhu cầu trao đổi thông tin trên mạng Internet, người ta xây dựng một kết nối có tốc độ cực nhanh đó là các tuyến *backone*, là các siêu xa lộ sử dụng loại cáp quan để truyền dữ liệu với tốc độ lên tới 622 megabits/s.

+ *Mạng Intranet, Extranet và Internet :* Khi bạn xây dựng một mạng LAN, MAN hoặc WAN theo chuẩn Internet thì bạn đã tạo ra một mạng Intranet. Khi bạn kết nối mạng Intranet vào Internet và bắt đầu giao tiếp với thế giới bên ngoài bạn đã tạo ra một Extranet.

IV.Các ứng dụng Client/Server trên Internet thông dụng :

Tuỳ theo thông tin lưu trữ và mục đích phục vụ mà các Server trên mạng Internet sẽ được phân chia thành các loại như Web server, Email server, Chat Server, hay FPT server,….. mỗi loại sẽ được tối ưu hoá theo mục đích sử dụng giao thức và cổng kết nối khác nhau.

1. World Wide Web(www):

+ Web là một ứng dụng khá hoàn hảo và phổ biến nhất hiện nay, và ngày nay nó cấu thành phần lớn nhất của Internet dựa trên kỹ thuật biểu diễn thông tin gọi là siêu văn bản, trong đó các từ được chọn trong văn bản có thể được mở rộng bất cứ lúc nào để cung cấp đầy đủ hơn thông tin về từ đó. Sự mở rộng ở đây theo nghĩa là chúng có thể liên kết tới các tài liệu khác: văn bản, hình ảnh, âm thanh, hay hỗn hợp các loại….có chứa thông tin bổ sung. Nói cách khác World Wide Web là phần đồ hoạ của Internet. Thuở ban đầu, Internet là hệ thống truyền thông Internet là hệ thống truyền thông dựa trên văn bản; việc liên kết với những site khác có nghia là phải gõ những địa chỉ mã hoá dài dằng dặc với độ chính xác 100%. Công nghệ World Wide Web xuất hiện như là một vị cứu tinh. Khả năng đặt hình ảnh lên Web Site bất ngờ làm cho thông tin trên Web trở nên hấp nên hơn, lôi cuốn hơn. Ngoài ra HTTP (Hypertext Transfer Protocol) cho phép trang Web kết nối với nhau qua các siêu liên kết (hyperlink), nhờ vậy mà người dùng dễ dàng "nhảy" qua các Web site nằm ở hai đầu trái đất, World Wide Web chỉ là một phần cấu thành nên Internet ngoài ra còn có rất nhiều thành phần khác như: E-mail, Gopher, Telnet, Usenet... Các trình duyệt ở các máy Client sẽ thay mặt người sử dụng yêu cầu những tập tin HTML từ Server Web bằng cách thiết lập một kết nối với máy Server web và đưa ra các yêu cầu tập tin đến Server. Server nhận những yêu cầu này, lấy ra những tập tin và gởi chúng đến cửa sổ của trình duyệt ở Client.

+ Web Server là web cung cấp thông tin ở dạng siêu văn bản, được biểu diễn ở dạng trang. Các trang có chứa các liên kết tham chiếu đến các trang khác hoặc đến các tài nguyên khác trên cùng một Web Server hoặc trên một Web Server khác. Các trang tư liệu siêu văn bản sau khi soạn thảo sẽ được quản lý bởi chương trình Web Server chạy trên máy Server trong hệ thống mạng.

Cơ chế hoạt động của Web server

+ Máy server Web dùng giao thức HTTP để lấy tài nguyên Web xác định thông qua URL. HTTP là một giao thức mức ứng dụng được thiết kế sao cho truy cập tài nguyên Web nhanh chóng và hiệu quả. Giao thức này dựa vào mô hình request-reponse. Dịch vụ Web xây dựng theo mô hình client/server, trong đó Web browser đóng vai trò là client gởi các yêu cầu dưới dạng URL đến server. Web server trả lời bằng cách trả về một trang Hypertext Markup Language (HTML).

+ Trang HTML có thể là một trang tĩnh, tức là nội dung của nó đã có dạng xác định và được lưu trên Web site, hoặc một trang Web động (nội dung không xác định trước) mà server tạo ra tại thời điểm client yêu cầu để trả lời cho yêu cầu của client, hoặc một trang liệt kê các file và folder trên Web site.

Hình 1.3 : Web browser gửi yêu cầu URL đến Web server

+ Web browser gửi yêu cầu URL đến Web server

♦Mỗi trang trong một intranet hoặc trên Internet có một URL (Uniform Resource Location) duy nhất định vị chúng. Web browser yêu cầu một trang bằng cách gửi một URL đến một Web server. Web server sẽ dùng các thông tin trong URL để định vị và tổ chức một trang HTML để gửi về cho Web browser.

♦ Một chuỗi URL nói chung có dạng sau:

<protocol>://<domain_name of Server>/<path>

Trong đó:

✓ Tiền tố <protocol> chỉ ra giao thức được sử dụng cho dịch vụ, ví dụ giao thức Hypertext Transport Protocol (HTTP) được dùng cho dịch vụ Web, giao thức FTP, gopher,.....

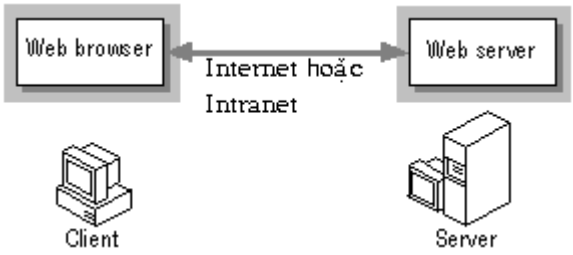
✓ <domain_name of Server> là tên DNS (Domain Name System) của máy Web server.

✓ <path> là đường dẫn đến thông tin được yêu cầu trên server.

Bảng sau ví dụ về các địa chỉ URL khác nhau:

Protocol	Domain name	Path
http://	www.hcmuns.edu.vn	/vanphong/dtao.htm
https:// (secure HTTP)	www.company.com	/catalog/orders.htm
gopher://	gopher.college.edu	/research/astronomy/index.htm
ftp://	orion.bureau.gov	/stars/alpha quadrant/startlist.txt

+ Web server trả lời yêu cầu của Web browser



Hình1.4 : Web server trả lời yêu cầu URL đến Web browser

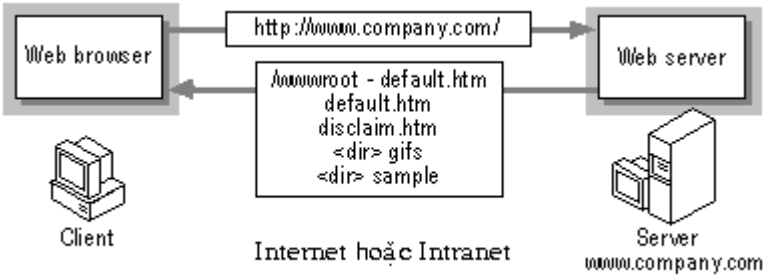
♦ Web server sẽ trả một trang HTML về cho Web browser, các trang HTML thuộc một trong 3 kiểu sau:

✓ Trang Web tĩnh (Static webpage) : là những trang HTML được chuẩn bị sẵn. Web server chỉ đơn giản là lấy trang này gởi về cho Web browser mà không gọi thi hành một chương trình hay một script nào khác. người dùng yêu cầu một trang Web tĩnh bằng cách nhập vào một chuỗi URL hoặc click chuột vào một siêu liên kết trỏ tới URL.

✓ Trang Web động (Dynamic webpage) : là những trang Web được tạo ra tại thời điểm client gửi yêu cầu để đáp ứng yêu cầu của user. Server có thể sẽ gọi chạy một chương trình khác, sử dụng các API của

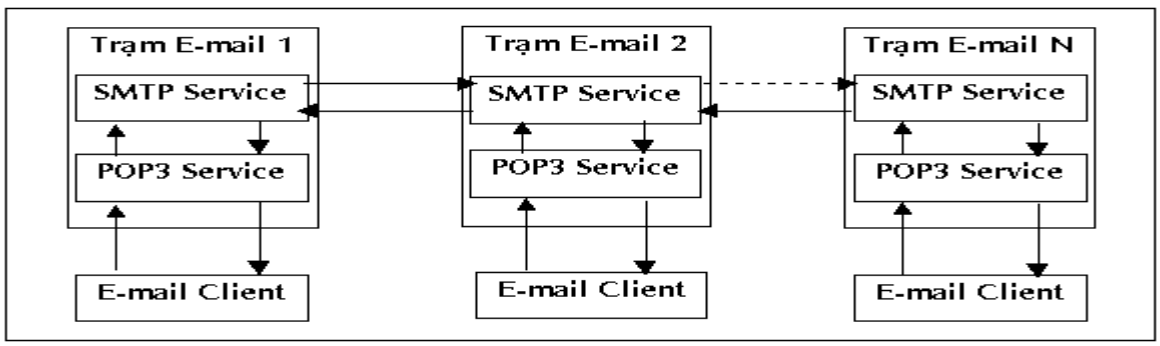
server, các ngôn ngữ kịch bản CGI script, query cơ sở dữ liệu ... tuỳ theo các thông tin mà web browser cung cấp.

✓ Danh sách liệt kê(Directory listing) : Nếu user gửi yêu cầu mà không mô tả một file cụ thể, thì có thể tạo một trang mặc nhiên cho Web site hay cho một thư mục, hoặc cấu hình server cho phép duyệt thư mục. Nếu sử dụng trang HTML mặc nhiên cho thư mục, thì trang này sẽ được gửi cho Web browser, còn nếu không có thì một directory listing (phiên bản HTML của Windows Explorer hay File Manager chạy trên trình duyệt) được trả về cho user dưới dạng một trang HTML, trong đó mỗi file và thư mục thể hiện như một siêu liên kết. Sau đó user có thể nhảy đến một file bất kỳ bằng cách click vào siêu liên kết tương ứng trong directory-listing.



2. <u>Thư điện tử (E-Mail):</u>

- Là dịch vụ rất phổ biến và thông dụng trong mạng Internet/Intranet và hầu như không thể thiếu được trong Internet/Intranet hiện nay. Tuy nhiên không phải là dịch vụ "*từ đầu - đến cuối*" (end to end). Nghĩa là dịch vụ này không đòi hỏi hai máy tính gởi và nhận thư phải nối trực tiếp với nhau để thực hiện việc chuyển thư. Nó là dịch vụ kiểu lưu và chuyển tiếp (store and forward) thư được chuyển từ máy này sang máy khác cho tới khi máy đích nhận được. Người nhận cũng chỉ thực hiện một số thao tác đơn giản để lấy thư, đọc thư và nếu cần thì cho in ra. Cách liên lạc này thuận tiện hơn nhiều so với gởi thư thông thường qua bức điện hoặc Fax, lại rẻ và nhanh hơn. Cách thực hiện việc chuyển thư không cần phải kết nối trực tiếp với nhau để chuyển thư, thư có thể được chuyển từ máy này đến máy khác cho tới máy đích.. Giao thức truyền thống sử dụng cho hệ thống thư điện tử của Internet là SMTP(Simple Mail Transfer Protocol). Cơ chế hoạt động của thư điện tử(E-mail):



- Giao thức liên lạc : mặc dù gởi thư trên Internet sử dụng nhiều giao thức khác nhau, nhưng giao thức SMTP (Single Message Transfer Protocol) được dùng trong việc vận chuyển mail giữa các trạm. Giao thức này đặc tả

trong hai chuẩn là trong RFC 822 (định nghĩa cấu trúc của thư) và RFC 821(đặc tả giao thức trao đổi thư giữa hai mạng) ngoài ra trong rfc2821 sẽ nói rõ các qui luật và cách hoạt động của giao thức. Là giao thức cơ bản để chuyển thư giữa các máy Client, SMTP có một bộ gởi thư, một bộ nhận thư, và một tập hợp lệnh dùng để gởi thư từ người gởi đến người nhận. Giao thức SMTP hoạt động theo mô hình khách/chủ (Client/ Server) với một tập lệnh đơn giản, trình khách (SMTP mail Client) sẽ bắt tay với trình chủ (SMTP mail Server) gởi các yêu cầu tiếp nhận mail. Trình chủ đọc nội dung mail do trình khách gởi đến và lưu vào một thư mục nhất định tương ứng với từng user trên máy chủ. Phần này sẽ được làm rõ hơn trong nhưng chương sau.

- Cứ mỗi trạm e-mail thường bao gồm ít nhất là hai dịch vụ: *POP3 (Post Office Protocol Version 3)* có nhiệm vụ nhận/trả thư từ/tới e-mail client và dịch vụ *SMTP (Simple E-mail Transfer Protocol)* có nhiệm vụ nhận/phân phối thư từ/đến POP3 đồng thời trao đổi thư với các trạm e-mail trung gian. POP3 được tìm thấy trong rfc1725 hay RFC 1939, là một giao thức đơn giản nhất, cho phép lấy mail về từ trình chủ POP3 Server. Ngoài tra trạm e-mail này có thể bổ sung thêm một số dịch vụ khác như ESMTP, IMAP và dịch vụ MX Record của dịch vụ DNS hay dịch vụ chuyển tiếp mail(Forward or relay). IMAP(INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1) thực chất là giao thức mới bổ Sung và mở rông hơn của giao thức POP3 còn thiếu. IMAP cho phép đọc, xoá, gởi, duy chuyển mail ngay trên máy chủ. Điều này rất thuận tiện cho người nhận mail phải thường xuyên di chuyển mail từ máy này sang máy khác trong quá trình làm việc. Tuy nhiên chi phí để cài đặt một trạm e-mail có giao thức IMAP là rất cao so với giao thức POP3.

- Mỗi người dùng (client) đều phải kết nối với một E-mail Server gần nhất (đóng vai trò bưu cục địa phương) phải có một tên (e-mail account) trên một trạm e-mail và sử dụng chương trình e-mail client (ví dụ như Eudora, Netscape...). Sau khi soạn thảo xong thư và đề rõ địa chỉ đích (người nhận) rồi gửi thư tới E-mail-Server của mình. E-mail Server này có nhiệm vụ sẽ tự động kiểm tra và định hướng chuyển thư tới đích hoặc chuyển thư tới một E-mail-Server trung gian khác. Thư chuyển tới E-mail-Server của người nhận và được lưu ở đó. Đến khi người nhận thiết lập tới một cuộc kết nối tới E-mai-Server đó thì thư sẽ chuyển về máy người nhận, nếu không thì thư vẫn tiếp tục giữ lại ở server đảm bảo không bị mất.

- Phần khác của ứng dụng thư điện tử là cho phép người sử dụng đính kèm (attachments) theo thư một tập tin bất kỳ (có thể dạng nhị phân chẳng hạn chương trình chạy). E-mail đã và đang hết sức thành công đến nỗi những người sử dụng Internet phục vụ dùng nó đối với hầu hết các trao đổi của họ. Một lý do làm e-mail Internet phổ biến là vì việc thiết kế nó rất cẩn thận: giao thức làm cho việc "phát thư" có độ tin cậy cao. không chỉ hệ thống thư tín trên máy của người gởi tương tác trực tiếp trên máy của người nhận mà giao thức còn đặc tả một thông điệp không thể bị xoá bởi người gởi cho đến khi người nhận đã thật sự có một phiên bản của thông điệp trên bộ lưu trữ (đĩa cứng chẳng hạn)của họ.

- Như vậy để gởi/nhận thư người sử dụng chỉ cần quan tâm tới cách sử dụng chương trình e-mail client. Hiện nay có nhiều chương trình e-mail client như Microsoft Outlook Express, Eudora Pro, Peagasus mail,....

3. <u>Dịch vụ Chat:</u>

- Chat là tài nguyên được mọi người sử dụng trên Internet ưa chuộng nhất. Đây là tài nguyên rất lý thú, nó cho phép bạn thiết lập các cuộc đối thoại thông qua máy vi tính với người dùng khác trên Internet. Sau khi bạn đã thiết lập được hệ thống này, những gì bạn gõ trên máy tính của bạn gần như tức thời trên máy tính kia và ngược lại. Những cuộc trao đổi thông qua chương trình Chat là sự đối mặt trực tiếp giữa hai người đối thoại với nhau thông qua ngôn ngữ viết nên sẽ chậm hơn so với đối thoại bằng miệng nhưng chỉ có lợi ích nhất là với những người không cùng ngôn ngữ vì gõ-đọc dễ hơn nghe-nói và trong một số trường hợp khác thì gõ(viết) dễ hơn là nói.

4. Dịch vụ FPT (File Transfer Protocol)

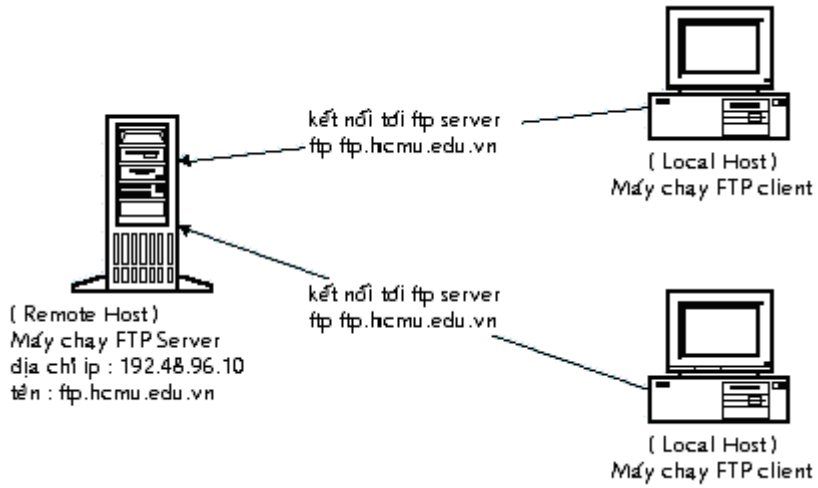
- Là dịch vụ truyền tập tin(tệp) trên Internet. FPT cho phép dịch chuyển tập tin từ trạm này sang trạm khác, bất kể trạm đó ở đâu và sử dụng hệ điều hành gì, chỉ cần chúng đều được nối với Internet và có cài đặt FPT. FPT là một chương trình phức tạp vì có nhiều cách khác nhau để xử lý tập tin và cấu trúc tập tin, và cũng có nhiều cách lưu trữ tập khác nhau.
- Để khởi tạo FPT từ trạm làm việc của mình người sử dụng chỉ gõ :

Fpt<domain name or IP address>

- Fpt sẽ thiết lập liên kết các trạm xa và bạn sẽ đăng nhập vào hệ thống(login/password). Vì fpt cho phép truyền tập tin theo cả hai chiều. Để chuyển tập tin của mình đến trạm ở xa dùng lệnh put, và ngược lại dùng lệnh get để lấy thông tin về. Ngoài ra trong một số trường hợp nó có thể đổi tên, tạo, xoá thư mục….FPT Client sử dụng dịch vụ để lấy(get) các tập tin từ FPT Server về máy của mình (download) hoặc gởi(put) các tập tin lên FPT server (upload).

```
ftp>put  source-file  destination-file
ftp>get  source-file  destination-file
```

- FTP theo nghĩa tiếng việt là nghi thức truyền file giữa các máy tính này đến máy tính khác thông qua mạng. Nếu như nghi thức TCP/IP gồm có các lớp Application, lớp TCP, lớp IP, lớp Network, lớp Datalink và lớp Physical thì FTP thuộc lớp ứng dụng (Application).
- WWW là một dịch vụ hấp dẫn, nó thay thế hầu hết những chức năng của FTP. Tuy nhiên chỉ có FTP mới cho phép copy file từ máy tính Client đến Server. Nếu một người dùng từ xa muốn làm điều này thì chắc chắn họ phải dùng FTP. Những loại file có thể truyền được bằng FTP rất phong phú, từ các file tư liệu(document) cho đến các file Multimedia như file hình ảnh, âm thanh.



Hình 1.5 Mô hình truyền nhận File FPT

Người sử dụng chương trình fpt Client kết nối với fpt Server, để kết nối thành công người dùng phải biết địa chỉ IP hoặc tên của máy chủ chạy fpt Server được gọi là trạm ở xa(Romote host) và máy chạy fpt Client được gọi là trạm địa phương(local host), thường thì chúng ta(người sử dụng) chỉ sử dụng chương trình fpt Client.

5. Đăng nhập từ xa Telnet

- Telnet là một chương trình dùng giao thức Telnet, nó là một phần của bộ giao thức TCP/IP. Nó cho phép người sử dụng từ một trạm làm việc của mình có thể đăng nhập vào một mạng ở xa qua mạng và làm việc với hệ thống y như một trạm cuối nói trực tiếp với trạm ở xa đó.

- Máy tính ở xa, còn được gọi là telnet, sẽ chấp nhận nối kết telnet từ một máy tính trên một hệ thống TCP/IP. bởi vì Internet là một mạng TCP/IP, telnet sẽ làm việc một cách hài hoà giữa các máy tính nối đến nó nếu như dịch vụ telnet được cài đặt trên máy tính của bạn. các thành phần telnet và server thoả thuận trong cách mà chúng sẽ dùng kết nối, vì thế mặc dù các hệ thống không cùng loại chúng vẫn tìm thấy một ngôn ngữ chung. telnet cũng có những giới hạn của nó, nếu lưu thông trên mạng kết nối từ xa có thể khiến cho sự cập nhật từ màn hình trở nên chậm hơn. Telnet thường dùng cho các mục đích công cộng và thương mại, cho phép những người dùng ở xa tìm kiếm các cơ sở dữ liệu lớn, phức tạp, và nó cũng là nguồn tài nguyên có giá trị trong giáo dục giúp cho việc nghiên cứu của bạn trở nên hấp dẫn hơn.

- Để khởi động Telnet, từ trạm làm việc của mình người sử dụng chỉ việc gõ:

```
telnet <domain-name or IP-address>
```

Người sử dụng kết nối đến Server Telnet(thường gọi là daemon) sẽ sử dụng cổng 23 cho những kết nối đến Server. Để hiểu rõ việc truyền thông giữa Telnet Client và Telnet Server thì bộ RFC 854 nói lên mối liên lạc này. RFC 854 xác định được 3 thành phần cơ bản trong bộ giao thức Telnet.

- Khái niệm thiết bị đầu cuối ảo(Network Virtual Terminal).
- Những qui tắc tuỳ chọn cho việc dàn xếp để chuyển dữ liệu.
- Sự tương xứng giữa thiết bị đầu cuối và các tiến trình.

6. Archie (tìm kiếm tập tin)

Phát triển tại đại học McGill ở Canada, Archie là một loại thư viện khổng lồ sẽ tự động và đều đặn tạo ra một số lớn các thông tin gởi đến máy chủ trên Internet và lập chỉ mục các tập tin của chúng để tạo ra một cơ sở dữ liệu duy nhất có thể tìm kiếm được. CSDL này còn là mục lục của dữ liệu danh mục, một sự biên dịch các tập tin có sẵn trên mọi máy chủ, Archie quét qua các máy chủ Internet một cách thường xuyên, và CSDL này thường xuyên được cập nhật. thực sự thì Archie không phải là một hệ thống độc lập, thay vì vậy nó là một nhóm các máy chủ. mỗi máy chủ archie đáp ứng cho sự tra hỏi các máy chủ Internet của chính nó để tạo nên cơ sở dữ liệu cho chính nó.

7. Gopher(Dịch vụ tra cứu thông tin theo thực đơn)

Gopher cho phép ta truy cập vào nhiều nguồn tài nguyên khác nhau, nhiều loại dịch vụ của Internet. Là một hệ thống làm việc theo Client/Server dưới dạng thực đơn(Menu), có thể duy chuyển từ menu này sang menu khác. Nếu thông tin cần tìm không có ở trạm kết nối thì Gopher Server sẽ tự động nối đến trạm khác.

Hệ thống Gopher phát triển bởi đại học Minnesota và được miễn phí cho các hoạt động phi lợi nhuận, Gopher có thể được dùng trên một số hệ thống máy tính như: UNIX, DOS, Microsoft Windows, Macintosh, OS/2...Phần mềm Client chạy trên máy tính của bạn có thể chạy trên bất kỳ máy nào của Gopher. Với Gopher bạn có thể đi xuyên qua Internet và đi đến những nơi mà không có người dùng nào đã từng đi đến, cách mà nó thực hiện bởi tổng hợp các công cụ Internet như: Telnet, FPT, để khi bạn tìm ra một đề mục tương quan đến những gì bạn đang tìm kiếm, bạn có thể đi trực tiếp đến nó mà không cần một trình tiện ích, hãy nhập vào địa chỉ của mục tiêu việc tìm kiếm....Gopher sẽ lấy tất cả điều này cho bạn.

8. Tìm kiếm thông tin theo chỉ số (WAIS)

Cũng giống như Gopher, WAIS(Wide Area Information Server) cho phép tìm kiếm và truy cập thông tin trên mạng(phần lớn là thông tin văn hoá) mà không cần biết chúng đang thực sự ở đâu. WAIS cũng hoạt động theo mô hình Client/Server, tuy nhiên ngoài WAIS Client và WAIS Server còn thêm WAIS indexer thực hiện việc cập nhật dữ liệu mới, sắp xếp theo chỉ số để tiện trong việc tìm kiếm. WAIS không chỉ cho phép hiển thị tập tin văn bản mà còn những tập tin đồ hoạ. Nó là nguồn quan trọng giúp cho các nguồn thông tin trên Internet có thể truy xuất được.

WAIS là một trong những chương trình đầu tiên dựa vào tiêu chuẩn Z39.50(tiêu chuẩn của American National Standard), nó là hệ thống đầu tiên dùng tiêu chuẩn này, nó trở thành một dạng thức tìm kiếm phổ biến, WAIS có thể nối đến bất kỳ CSDL hoặc máy Client có dùng Z39.50.

9. Dịch vụ tên miền (Domain Name System - DNS)

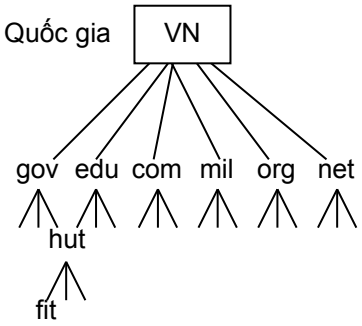
- Việc định danh các phần tử của liên mạng bằng các con số như trong địa chỉ IP rõ ràng là không làm cho người sử dụng hài lòng, bởi chúng khó nhớ, dễ nhầm lẫn. Vì thế người ta đã xây dựng hệ thống đặt tên (name) cho các phần tử của Internet, cho phép người sử dụng chỉ cần nhớ đến các tên

chứ không cần nhớ đến các địa chỉ IP nữa. Ta có thể biết thêm thông tin cách hoạt động của dịch vụ này thông qua RFC 1035.

- Hệ thống này được gọi là DNS (Domain Name System). Đây là một phương pháp quản lý các tên bằng cách giao trách nhiệm phân cấp cho các nhóm tên. Mỗi cấp trong hệ thống được gọi là một miền (domain), các miền được tách nhau bởi dấu chấm. Số lượng domain trong một tên có thể thay đổi nhưng thường có nhiều nhất là 5 domain. Domain có dạng tổng quát là *local-part@domain-name.*

trong đó :

- *Local-part* thường là tên của một người sử dụng hay nhóm người sử dụng do người quản lý mạng nội bộ qui định.
- Còn *domain-name* được gán bởi các Trung tâm thông tin mạng (NIC) các cấp. Domain cấp cao nhất là cấp quốc tế(com, org, net,..) sau đó là cấp quốc gia và mỗi quốc gia được gán một tên miền riêng biệt gồm hai chữ cái. Ví dụ *vn* (Việt Nam), *us* (Mỹ), *ca* (Canada), *fr* (Pháp), v.v...Trong từng quốc gia lại được chia thành 6 domain cao nhất và tiếp tục đi xuống các cấp thấp hơn.



Domain	Phạm vi sử dụng
Gov	các tổ chức chính phủ (phi quân sự)
Edu	các cơ sở giáo dục
Com	các tổ chức kinh doanh, thương mại
Mil	các tổ chức quân sự
Org	các tổ chức khác
Net	các tài nguyên mạng

- Mỗi một Domain cấp chính cần phải cung cấp cho một DNS Server, DNS s Server này có nhiệm vụ lưu trữ địa chỉ các Domain con của nó nhằn mục đích giúp người sử dụng tìm kiếm và truy xuất vào các địa chỉ này một cách dễ dàng. Các DNS Server đều liên lạc được với nhau.

10. Dịch vụ nhóm tin (Use Net News Groups)

Là dịch vụ cho phép nhiều người ở nhiều nơi khác nhau có thể tham gia công tác hay trao đổi về một chủ đề riêng nào đó hoặc những người có cùng mối quan tâm giống nhau có thể tham gia vào một nhóm tin để trao đổi về vấn đề đó. Mỗi chủ đề được thảo luận trong một nhóm riêng biệt. Chủ đề của một nhóm trong một nhóm riêng biệt. Chủ đề của một nhóm tin thì vô cùng phong phú ví dụ như: nhóm tin thuộc nhạc cổ điển, nhóm tin về thể thao, nhóm tin khoa học….. Xoay quanh mọi vấn đề trong cuộc sống, có thể nói không có vấn đề gì không có trong nhóm tin, mỗi nhóm tin có thể có nhiều nội dung thảo luận. Khi bạn gởi một bản tin đến một nhóm tin chủ thì chủ đó sẽ tiếp tục gởi bản tin đến một nhóm chủ cùng cộng tác trên Internet, và thông tin có thể lấy từ các Server (máy chủ) khác nhau. Vì vậy những người khác có thể lấy về và đọc bản tin đó từ News Server mà họ nối tới. Việc gởi bản tin tới nhóm tin

cũng tương tự như E-mail chỉ khác ở chỗ là địa chỉ gởi là địa chỉ của nhóm tin và việc lấy các văn bản về đọc cũng tương tự như lấy và đọc E-mail. Và người sử dụng cũng chỉ cần biết đến một server tin duy nhất, đó là server tin mà mình kết nối vào. Mọi sự trao đổi, tương tác giữa các server tin và các nhóm tin là hoàn toàn trong suốt đối với người sử dụng. Với dịch vụ này, người sử dụng có thể nhận được thông tin cần thiết từ nhiều người từ khắp thế giới.

CHƯƠNG 2

KIẾN TRÚC MẠNG VÀ CÁC PROTOCOL TRUYỀN THÔNG MẠNG

I.Kiến trúc mạng

Có thể chia cấu trúc mạng làm hai phần như sau:

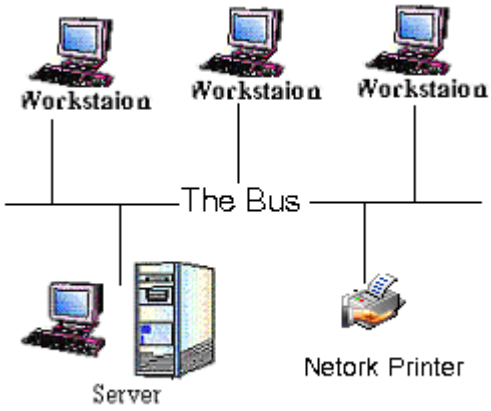
+ Phần vật lý: gồm tất cả những gì liên quan đến phần cứng như máy tính, dây cáp mạng, card mạng và các thiết bị khác để truyền dữ liệu trên mạng.

+ Phần lôgic: là cách tổ chức lôgic của các thiết bị phần cứng nói trên để chúng hiểu và làm việc với nhau.

1. kiến trúc vật lý:

Các máy tính được kết nối với nhau thông qua cáp mạng và card mạng(NIC: Network Interface Card) được lắp đặt cho từng máy. Nhiệm vụ của NIC làm cho máy tính có thể giao tiếp được với các thiết bị khác trên mạng. Hiện nay có 3 kiểu cấu hình mạng thông dụng là mạng vòng(bus topolopy), mạng sao(star topolopy) và mạng vòng(ring topolopy). Cấu hình hus, star thường được dùng trong mạng Ethernet, mạng vòng được dùng trong mạng Token Ring.

+ *Mạng bus :* có ưu điểm là cấu hình đơn giản, khi các máy nối vào hệ thống mạng thì cần cài đặt phần mềm cho mỗi máy tính là có thể sử dụng được, các máy này nhận được máy kia dễ dàng. Nhược điểm là có quá nhiều yếu điểm trên đường truyền, chỉ cần mối kết nối giữa hai máy nào đó bị trục trặc là toàn bộ hệ thống mạng điều chết.



Hình 2.1 Mạng cấu hình bus Ethernet 10BASE2

+ *Mạng sao:* hệ thống cáp mạng nối lần lượt từ máy này sang máy khác ở dạng hình sao, người ta sử dụng một thiết bị làm trung tâm kết nối chung cho tất cả các máy gọi là hub(Switch,…). Thiết bị này có nhiệm vụ điều phối tất cả giao tiếp giữa các máy trên mạng.

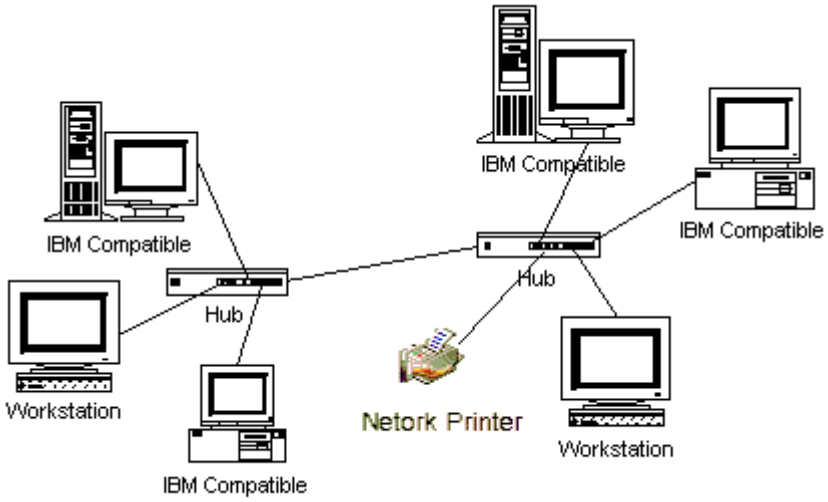
Ưu điểm :

- Dễ phát hiện những sự cố về đường dây cáp kết nối.
- Nếu có sự cố về đường dây không ảnh hưởng đến toàn bộ hệ thống.
- Lưu lượng dữ liệu trên đường dây ít đụng độ nhờ có các thiết bị kết nối chuyên dùng.

- Có thể giảm bớt hoặc thêm máy kết nối mạng mà không ảnh hưởng đến hệ thống mạng.

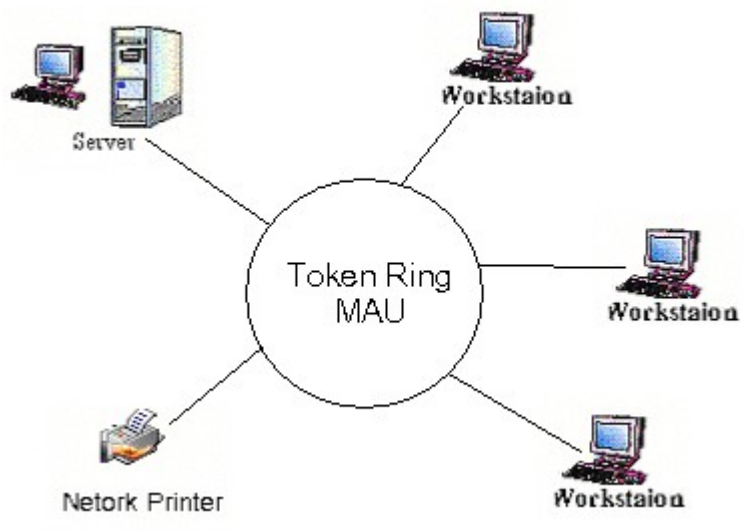
Khuyết điểm :

- Chi phí cho cáp kết nối cao.
- Các đầu nối tập trung tại một vị trí, quản lý phức tạp.



Hình 2.2 Mạng sao Ethernet 10BASE_T

+ *Mạng vòng:* được dùng với mạng Token Ring hoặc FDDI cách tổ chức hệ thống thiết bị phần cứng giống như mạng sao nhưng không sử dụng hub hay switch mà thay vào đó bằng thiết bị trung tâm gọi là MAU(Multistation Access Unit). Các hoạt động của MAU cũng tương tự như hub hay switch nhưng nó được sử dụng trong mạng Token Ring.



Hình 2.3 Mạng Token Ring

2. <u>Kiến trúc logic mạng:</u>

Là tập hợp các tài nguyên như đĩa cứng, máy in, các ứng dụng đang chạy trên mạng hay có thể nói kiến trúc lôgic mạng là thuật ngữ chỉ sự tổ chức mạng. hay nói cách khác sự tổ chức các phần cứng mạng được thực hiện bởi phần mềm mạng sẽ tạo ra cấu trúc lôgic mạng.

II.Truyền thông mạng và kiến trúc phân tầng của protocol:

1. Truyền thông mạng:

Yếu tố quan trọng của mạng máy tính là tập hợp các máy tính được nối với nhau bởi các đường truyền và theo kiến trúc của một mạng máy tính. Vậy các máy tính này được truyền thông với nhau ra sao, tập hợp các qui tắc, quy ước, cách truyền thông trên mạng phải tuân theo như thế nào để cho mạng hoạt động tốt. Cách nối các máy tính được gọi là hình trạng(Topolopy) của mạng. Còn tập hợp tất cả những qui tắc, qui ước truyền thông thì được gọi là giao thức(protocol) của mạng. Topolopy và Protocol là hai khái niệm cơ bản nhất của mạng máy tính.

- Topolopy có hai kiểu mạng chủ yếu là:

+ Kiểu điểm-điểm: các đường truyền nối từng cặp nút với nhau và mỗi nút đều có trách nhiệm lưu trữ tạm thời sau đó chuyển dữ liệu đi cho tới đích.

+ Kiểu truyền bá: Tất cả các nút phân chia chung một đường truyền vật lý. Nghĩa là dữ liệu được gởi đi từ một nút nào đó sẽ có thể được tiếp nhận bởi tất cả các nút còn lại.

- Protocol: phục vụ trong việc trao đổi thông tin, dù là cuộc trao đổi đơn giản nhất cũng phải tuân theo một qui tắc nhất định.Tập hợp tất cả những qui tắc, qui ước đó gọi lag giao thức(protocol) của mạng. Hiện nay có rất nhiều protocol mạng khác nhau nhưng thông dụng nhất vẫn là là giao thức TCP/IP. Vấn đề protocol được trình bày chi tiết hơn ở phần tiếp theo.

2. kiến trúc phân tầng và mô hình ISO của protocol:

a. kiến trúc phân tầng.

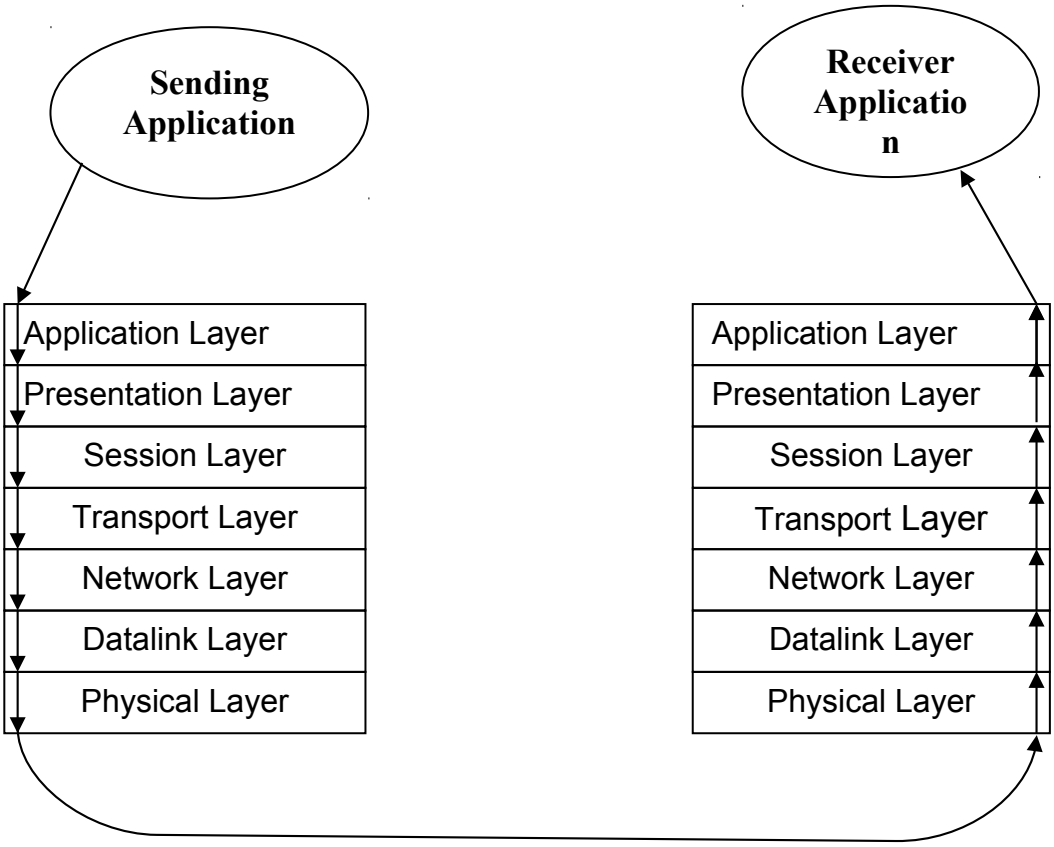
- Để có thể chuyển một thông điệp từ máy này sang máy khác(các máy phải dùng trong hệ thống mạng) nó phải trải qua nhiều giai đoạn khác nhau như là: chia nhỏ thông điệp ra thành nhiều gói nhỏ(package), mã hoá các gói này ra thành dạng bit, các bit này được chuyển qua đường truyền vật lý đến máy nhận. Sau đó quá trình nhận sẽ thực hiện ngược lại với bên gởi, nếu quá trình lắp ghép gặp phải lỗi thì phải thông báo để truyền lại,….Các giai đoạn này rất phức tạp đòi hỏi người lập trình phải hiểu rõ tất cả cơ chế hoạt động bên trong của hệ thống. Vì bậy người ta đưa ra ý tưởng phân tầng, mỗi tầng sẽ chịu trách nhiệm cung cấp dịch vụ cho tầng bên trên đồng thời nó cũng sử dụng dịch vụ của tầng bên dưới cung cấp cho nó. Như vậy thì một người làm việc ở tàng nào thì chỉ quan tâm đến tầng có quan hệ trực tiếp với mình.

- Để giảm độ phức tạp của việc thiết kết và cài đặt mạng, hầu hết các máy tính hiện có được thiết kế theo quan điểm phân tầng. Mỗi hệ thống thành phần của mạng được xem như là một cấu trúc đa tầng, trong đó mỗi tầng được xây dựng trên tầng trước đó. Số lược mỗi tầng cũng như tên hay các chức năng phụ thuộc vào nhà thiết kế. Chúng ta thấy cách phân tầng trong mạng IBM(SNA), mạng Digital(DECnet), hay bộ quốc phòng mỹ(ARPANET),.. là giống nhau. Mmặc dù tên và chức năng từng tầng là khác nhau giữa các mạng trên nhưng bản chất vẫn dựa theo mô hình phân tầng ISO.

b. Mô hình ISO.

- Khi thiết kế protocol các nhà thiết kết tự do chọn lựa cho lựa kiến trúc mạng riêng cho mình, từ đó dẫn tình trạng không tương thích mạng(phương

pháp truy cập đường truyền khác nhau, sử dụng họ giao thức khác nhau,….). Sự không tương thích đó làm trở ngại sự tương tác giữa người sử dụng với các mạng khác nhau một khi nhu cầu trao đổi thông tin ngày càng lớn thì sự trở ngại này không thể chấp nhận được. Sự thúc đẩy từ nhu cầu người dùng đã thúc đẩy các nhà sản xuất và nghiên cứu thông qua các tổ chức chuẩn hoá quốc gia và quốc tế tích cực tìm kiếm một sự hội tụ cho các sản phẩm mạng trên thị trường.Vì lý do đó, tổ chức chuẩn hoá quốc tế (Internationl Organization for Strandarization – viết tắt là ISO) đã xây dựng một mô hình protocol tham chiếu cho việc kết nối các hệ thống mở phục vụ cho các ứng dụng phân tán. Theo mô hình ISO, thông tin muốn gởi và nhận qua mạng phải đi qua 7 tầng. Mỗi tầng có một chức năng khác nhau và cung cấp các interface để tầng trên có thể sử dụng lớp dưới. Mô hình ISO được coi là mô hình chuẩn vì các mô hình khác cũng dựa theo mô hình này để tạo ra một mô hình phù hợp cho riêng mình, mà ngày nay thông dụng nhất là mô hình TCP/IP.



Hình2.4 Mô hình ISO gồm 7 tầng.

Giải thích

+ Physical: ở tầng này thông tin được truyền dưới dạng bit thông qua kênh truyền và nhận các bít chuyển tầng Datalink. Tầng này không có cấu trúc qua đường truyền vật lý, truy nhập đường truyền vật lý như các phương tiện cơ, điện, hàm, thủ tục.

+ Datalink: tầng này có nhiệm vụ chia nhỏ dữ liệu từ tầng Network đưa xuống thành các frame, mỗi frame có dung lượng vài trăm byte đến vài ngàn byte. Các frame được truyền đi bằng cách chuyển xuống cho tầng phisical. Nhiệm vụ thứ hai là tổ chức nhận các frame sao cho đúng thứ

tự, cung cấp khả năng truyền không lỗi trên đường truyền vật lý cho các lớp cao hơn.

+ Network: định hướng gói dữ liệu(package) đi từ máy gởi đến máy nhận. Phải giải quyết vấn đề định tuyến(routing), vấn đề địa chỉ(addressing), lượng giá chi phí(accouting), và giải quyết đụng độ(collision).

+ Transport:Chia nhỏ gói dữ liệu được đưa xuống từ tầng trên thành những đơn vị nhỏ hơn truyền qua mạng, với sự đảm bảo là dữ liệu đến nơi một cách chính xác.

+ Session: điều kiển quá trình giao tiếp giữa hai tuyến trình trên máy. Cung cấp phương tiện quản lý truyền thông giữa các ứng dụng, thiết lập duy trì đồng bộ hoá và huỷ bỏ các phiên truyền thông giữa các ứng dụng.

+ Presentation: biểu diễn những thông tin được truyền(được hiểu là cú pháp và ngữ nghĩa) nó đồng nhất các thông tin giữa các hệ thống khác nhau.

+ Application: Cung cấp các phương tiện để người sử dụng có thể truy nhập được vào môi trường ISO, đồng thời cung cấp các dịch vụ thông tin phân tán hay dịch vụ cho người sử dụng. Ứng với mỗi dịch vụ có một protocol khác nhau.

- Điều hấp dẫn của mô hình ISO chính là ở chỗ hứa hẹn giải pháp cho vấn đề truyền thông giữa các mạng không giống nhau. Hai hệ thống mạng dù khác nhau đi nữa điều có thể truyền thông với nhau một cách hiệu quả nếu chúng đảm bảo những điều kiện sau.

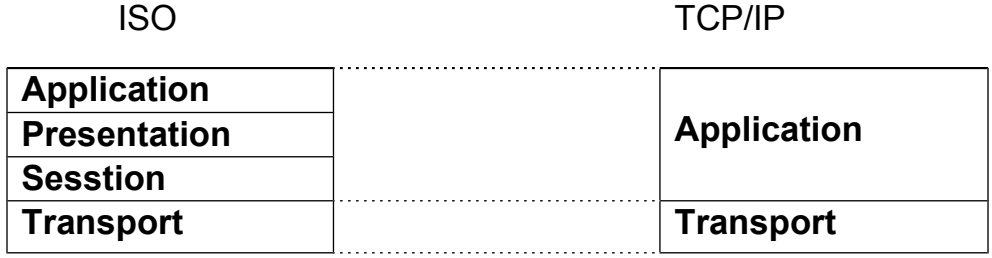
+ Chúng cài đặt cùng một tập các chức năng truyền thông.

+ Các chức năng đó được tổ chức cùng một tập các tầng. Các tầng đồng mức phải cung cấp các chức năng như nhau(phương thức cung cấp không nhất thiết phải giống nhau).

+ Các tầng đồng mức phải sử dụng chung một protocol.

c*. Mô hình TCP/IP*

- Chúng ta đã thấy được nguyên lý của mô hình ISO 7 lớp nhưng mô hình này chỉ là mô hình tham khảo, việc áp dụng mô hình ISO vào thực tế là khó có thể thực hiện được(hiệu suất kém vì dữ liệu khi truyền từ máy này sang máy khác trong mạng thì phải trải qua tất cả các lớp của mô hình ISO ở hai máy). Nó chỉ là tiêu chuẩn cho các nhà phát triển dựa theo đó mà phát triển thành các mô hình khác tối ưu hơn. Hiện nay có rất nhiều mô hình khác nhau trên mạng như SNA của IBM, DNA của DEC, TCP/IP của microsoft,… Tuy nhiên mô hình TCP/IP là được sử dụng phổ biến nhất hiện nay.



Network		**Internet**
Datalink		
Physical		**Host-to-network**

- Mô hình TCP/IP gồm 4 tầng, trong đó 2 tầng dưới của mô hình ISO được gộp lại thành 1 tầng gọi là Host-to-network, 2 tầng Sesstion và presentation không có trong mô hình TCP/IP.

- Tương tự như mô hình ISO, mô hình TCP/IP dữ liệu từ 1 máy cũng đi từ tầng Application xuống Transport rồi xuống tiếp tầng Internet sau cùng là Host-to-network thông qua đường vật lý đến một máy khác trên mạng: dữ liệu ở đây cũng đi ngược từ dưới lên như mô hình ISO. Chức năng và ý nghĩa từng tầng trong mô hình TCP/IP như sau:

+ *Host-to-network:* Đây là tầng giao tiếp mạng kết nối với network sao cho chúng có thể truyền các IP datagram tới các địa chỉ đích. Tầng này gần giống với tầng physical của ISO.

+ *Internet :*Thực hiện một hệ thống mạng có khả năng chuyển các gói dữ liệu dựa trên lớp mạng Connetionless(không cầu nối) hay Connection-Oriented(có cầu nối) tuỳ theo từng loại dịch vụ mà người ta dùng một trong hai cách trên.

+ *Transport :* được thiết kết cho các phần tử ngang cấp(hay host) có thể đối thoại với nhau thông qua một trong hai protocol sau đây.

TCP: là một Connection Oriented Protocol, cho phép chuyển một chuỗi byte từ host này sang host kia mà có thông báo trả về.

UDP: là một Connetionless protocol xây dựng cho các ứng dụng không muốn sử dụng cách truyền theo thứ tự của TCP mà muốn tự mình thực hiện điều đó và không có thông báo trả về nghĩa là nó không đảm bảo dữ liệu được truyền đi chính xác hay không.

- Một máy có thể liên lạc với một máy khác trong mạng qua địa chỉ IP(IP là danh từ dùng để định vị các host trên mạng). Tuy nhiên với một địa chỉ như vậy không đủ cho một process của máy này liên lạc với một process của máy khác. Vì vậy protocol TCP/UDP đã dùng một số nguyên(16 bit) để đặc tả nên số hiệu port liên lạc. như vậy mỗi fram của tầng Network bao gồm:

- Protocol(TCP/UDP).
- địa chỉ IP của máy gởi.
- Số hiệu port của máy gởi.
- địa chỉ IP máy đích.
- Số hiệu port máy đích.

+ *Application:* chứa các dịch vụ như trong các tầng Sesson, Presentation, Application của mô hình ISO như FPT(port=23), DNS(port=53), SPTP(port=25), IMAP(port=149),POP3(port=143),….

3.Giao thức TCP/IP

- Đầu tiên ARPANET đã đưa ra giao thức Host-to-Host Protocol, nhưng giao thức này không đáng tin cậy và nó chỉ giới hạn trong một số các máy. Vào cuối năm 1970 các mạng khác cũng bung ra trong thực tế, mạng UUCP gồm một nhóm rồi cũng đã nối được hàng trăm máy rồi hàng máy. Vào cuối năm 1980 mạng NSFNET mạng của National Science Foundation được phát triển để nối 5 trung tâm siêu máy tính của nó, nó là mạng hấp dẫn cho tất cả

các nhà nghiên cứu và các viện đại học cũng như các viện nghiên cứu. Năm 1972, bắt đầu thế hệ thứ hai của giao thức mạng, đã làm phát sinh ra một nhóm giao thức được gọi là Transmission Control Protocol/ Internet Protocol viết tắt là TCP/IP. Năm 1983, TCP/IP là bộ giao thức cho ARPANET, TCP/IP đã trở thành một trong những giao thức mạng được dùng rộng rãi nhất. Sau cùng tất cả các mạng được tài trợ bởi cá nhân hay xã hội -mạng ARPANET, MILNET, UUCP, BITNET, CSNET và NASA Science Internet đã liên kết trong một mạng khu vực NSFNET và ARPANET giải tán và ngày càng có nhiều mạng khác thêm vào...

- Ngày nay để thực hiện việc truyền thông qua mạng thông qua trình duyệt Web, và ta cũng cần một giao thức để thực hiện công việc này. Mặc dù hiện nay cũng đang có rất nhiều giao thức để truyền thông tin nhưng nhìn chung có hai giao thức thường được các lập trình viên sử dụng đó là: TCP/IP(IP: là giao thức Internet, TCP: giao thức truyền tải) và giao thức UDP(giao thức gói dữ liệu người dùng). Vì chương trình của em sử dụng giao thức TCP/IP nên sau đây em sẽ trình bày chi tiết giao thức này.

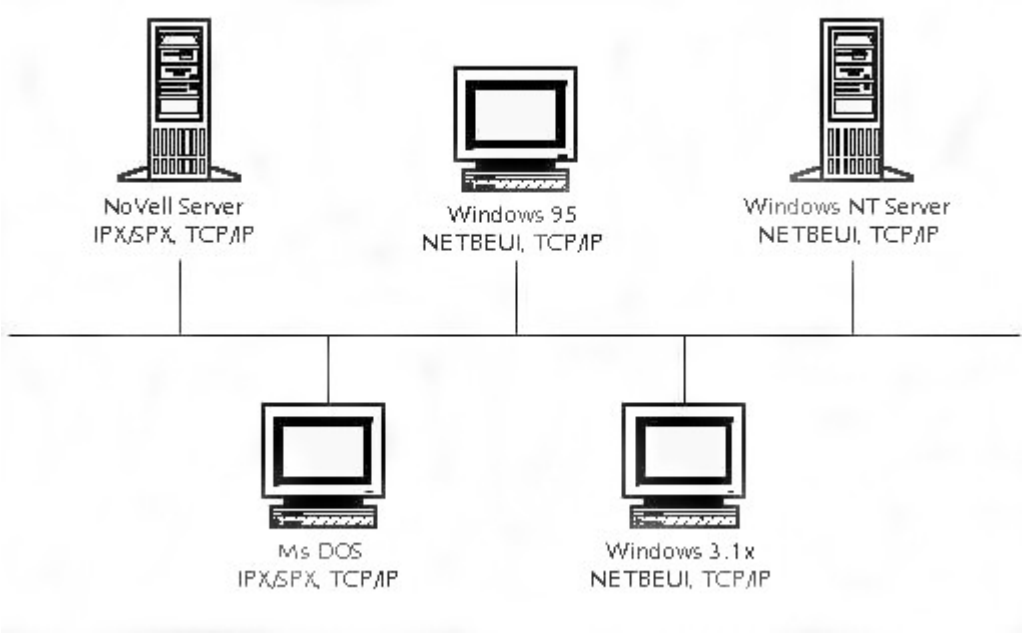
- Trong môi trường mạng máy tính dữ liệu trao đổi qua lại giữa các máy dựa trên nghi thức(Protocol), giao thức là cách đóng gói, mã hoá dữ liệu truyền trên đường mạng và các qui tắc thiết lập duy trì quá trình trao đổi dữ liệu. Như vậy, mặc dù có hai máy tính được kết nối về mặc vật lý trên cùng một đường truyền nhưng sử dụng hai nghi thức khác nhau cũng không trao đổi dữ liệu được. Hiện nay có nhiều nghi thức(protocol) được sử dụng nhưng chỉ có 3 giao thức phổ biến là:

+IPX/SPX : giao thức của hệ thống mạng Novell Netware.

+NETBEUI : giao thức chính của hệ thống mạng Microsoft Windows.

+TCP/IP: giao thức dùng cho hệ thống mạng Internet/Intranet /Extranet.

Tuy nhiên do sự bùng nổ của Internet/Intranet /Extranet các hệ mạng Novell Netware và Microsoft Windows cũng hỗ trợ và sử dụng thêm giao thức TCP/IP.



Hình 2.5: Ví dụ một mô hình mạng

Theo mô hình trên, các máy tính tuy sử dụng các hệ điều hành khác nhau nhưng lại chạy các phần mềm cùng hỗ trợ nghi thức TCP/IP nên có trao đổi dữ liệu qua lại với nhau dựa trên nghi thức này. Ngoài ra hai máy Server Novell và máy Ms DOS có thể dùng thêm giao thức IPX/SPX, các máy Windows có thể dùng thêm nghi thức NETBEUI để trao đổi dữ liệu với nhau. Như vậy, trên một máy tính có thể có nhiều cách thức khác nhau (sử dụng nhiều nghi thức khác nhau) để trao đổi dữ liệu với máy tính khác. Tuy nhiên, giao thức TCP/IP là phổ dụng nhất nghi thức chuẩn dùng cho Internet/Intranet/Extranet.

A. Các thành phần liên quan tới giao thức TCP/IP

1. Địa chỉ máy (IP Address)

- Mỗi nút (node - là một máy trạm, máy chủ hay bất kỳ thiết bị nào nối vào Internet) đều phải có phải có một địa chỉ duy nhất để phân biệt nó với các máy khác, và để tìm đường cho các packet trên mạng, gọi là địa chỉ IP.Địa chỉ IP là một chuỗi gồm có 4 số có giá trị từ 0 tới 255, phân cách giữa hai số là dấu chấm (.).

Ví dụ: 10.221.0.2, 130.23.1.17, 192.48.96.10 ...

- Tất cả các máy trong hệ thống mạng(LAN, WAN, Internet) đều có ít nhất 2 địa chỉ: địa chỉ vật lý(Mac Address) và địa chỉ Internet. Địa chỉ vật lý còn được gọi là Ethernet address là một dãy bit gồm 48 bit được gán bởi các nhà sản xuất, địa chỉ này được biểu diễn dưới dạng số thập lục phân (hecxa). Địa chỉ IP phải là duy nhất trên mạng và có một dạng thống nhất, mỗi địa chỉ IP gồm có 4 byte và có 2 thành phần: địa chỉ đường mạng (Network ID) và địa chỉ host(Host ID).

✓ Địa chỉ mạng: chỉ ra những máy, những thiết bị ở chung một vị trí trên mạng logic được chia theo Router (tất cả các máy trên cùng một phía của router thuộc chung một mạng logic).

✓ Địa chỉ máy: để phân biệt các máy trong một mạng logic. Mỗi máy trong một mạng logic phải có một địa chỉ máy duy nhất. Tuỳ thuộc vào giá trị của số thứ nhất mà địa chỉ IP được chia thành các lớp như A, B, C, D.

- Những máy trên mạng dùng Network ID và Host ID để quyết định xem nên nhận và bỏ qua các gói tin nào, và để quyết định phạm vi chuyển tin. Chỉ có các máy cùng Network ID mới nhận được các IP broadcast). Để biết gói tin đến có cùng Network ID với mình hay không, máy sẽ dùng Subnet mask của nó để tách địa chỉ IP của gói tin đến. Subnet mask là giá trị 32 bit, viết cách nhau bằng dấu chấm cho mỗi 8 bit. Subnet mask được gán các bit dành cho Network ID là 1 và các Host ID là 0. Bảng dưới là giá trị mặc định cho các lớp địa chỉ IP

Tên lớp	Subnet mask ở dạng bit	Dạng byte
Lớp A	11111111 00000000 00000000 00000000	255.0.0.0
Lớp B	11111111 11111111 00000000 00000000	255.255.0.0
Lớp C	11111111 11111111 11111111 00000000	255.255.255.0

Ví dụ : địa chỉ IP là 102.12.34.98 và subnet mask của nó là 255.255.0.0 thì Network Id của nó là 102.12 và Host ID là 34.98.

Nhìn thì có vẻ như subnet mask là thừa vì nhìn vào Network ID là có thể biết được các máy có cùng thuộc một mạng con hay không. Nhưng subnet mask còn dùng trong việc chia một mạng thành các mạng con (subnet).

- Một giải pháp giúp giảm nhẹ việc quản lý các địa chỉ IP, đó là giao thức tự động cấu hình và tự động cấp phát địa chỉ DHCP (Dynamic Host Configuration Protocol). DHCP dựa trên công nghệ Client/Server. Trng mạng có ít nhất một máy DHCP server có một khoảng địa chỉ dành để cấp phát cho các máy client. Các máy DHCP client khi khởi tạo sẽ tự động phát hiện máy DHCP server và yêu cầu máy chủ cấp cho một địa chỉ IP cùng các thông số cấu hình khác (subnet mask, địa chỉ gateway …). Máy server sẽ tự động cấp cho máy client một địa chỉ còn trống trong khoảng địa chỉ của nó. Khi máy client rời khỏi mạng, nó sẽ trả lại địa chỉ IP cho máy server.

- Địa chỉ IP là riêng biệt cho mỗi máy và là định danh của mỗi máy trong hệ thống mạng. Do vậy, để truy cập tới một máy bạn phải biết địa chỉ IP của nó. Tuy nhiên, vì địa chỉ IP thể hiện dưới dạng số nên thường khó nhớ, thông qua dịch vụ DNS (Domain *Name Service) cho phép đồng nhất một địa chỉ IP với một tên (thể hiện dưới* dạng chuỗi) và như vậy để truy cập tới một máy bạn có thể hoặc dùng địa chỉ IP hoặc dùng tên tương ứng với địa chỉ này.

2. Tên máy (Host name)

- Tên máy (host name) là sự đồng nhất giữa một tên với một địa chỉ IP. Tên máy đầy đủ bao gồm 2 phần: phần tên máy thuộc một miền và phần tên miền, giữa hai phần này phân cách nhau bởi dấu chấm (.) theo dạng host.[subdomain].domain.

Để quản lý các máy đặt tại những vị trí vật lý khác nhau trên hệ thống mạng nhưng thuộc cùng một tổ chức, cùng lãnh vực hoạt động... người ta đưa các máy này vào một *miền (domain). Trong miền này nếu có những tổ chức nhỏ hơn, lãnh vực hoạt động* hẹp hơn... thì lại được chia thành các miền con (sub domain), giữa hai tên miền phân cách nhau bởi dấu. Cấu trúc miền và các miền con giống như một cây phân cấp.

- Miền lớn nhất thường là cấp quốc gia, mỗi quốc gia có một tên miền gồm hai ký tự. Ví dụ: vn (Việt Nam), us (Mỹ), ca (Canada).... Trong miền mỗi quốc gia lại có các miền con như: edu (các tổ chức giáo dục), com (các tổ chức kinh doanh, thương mại)... Và cứ phân cấp xuống như thế mỗi miền con lại có nhiều miền con khác trong nó. Ví dụ miền hcmuns.edu.vn có nghĩa là miền con it-hut (đại học Bách Khoa Hà Nội) nằm trong miền con edu thuộc miền vn. Tên máy trong miền thường cũng được đặt tên theo chức năng hoạt động. Ví dụ như www để chỉ máy chạy dịch vụ World Wide Web, ftp để chỉ định tên máy chạy dịch vụ FTP....

Ví dụ : tên máy đầy đủ như: www.hcmuns.edu.vn, mail.hcmuns.edu.vn tương ứng với 2 máy có địa chỉ IP là: 172.29.2.154 và 172.29.2.155

- Để Kiểm tra sự tồn tại máy trong hệ thống mạng dùng giao thức TCP/IP dùng chương trình tiện ích có tên ping theo cú pháp như sau:

ping <địa_chỉ_IP | tên_máy>

Ví dụ như kiểm tra máy có địa chỉ 172.29.2.154(tên tương ứng www.hcmuns.edu.vn):

ping 172.29.2.154

ping www.hcmuns.edu.vn

- Nếu máy này có tồn tại trên hệ thống mạng thì sẽ có thông báo tương tự:

Pinging 172.29.2.154 with 32 bytes of data:
Reply from 172.29.2.154: bytes=32 time=1ms TTL=127
Reply from 172.29.2.154: bytes=32 time=1ms TTL=127
Reply from 172.29.2.154: bytes=32 time=1ms TTL=127
Reply from 172.29.2.154: bytes=32 time<10ms TTL=127

- Nếu máy này không có tồn tại thì sẽ có thông báo tương tự :

Pinging 172.29.2.154 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

B.Những TCP/IP protocols và các công cụ

- Như ta biết, truyền thông giữa hàng triệu computers trên Internet xảy ra được nhờ có TCP/IP protocol, một cách giao thức trên mạng rất thông dụng trong vòng các computers chạy Unix trước đây. Vì nó rất tiện dụng nên Microsoft đã dùng TCP/IP làm giao thức chính cho mạng Windows2000. TCP/IP là tập hợp của nhiều protocols, mà trong số đó có các Protocols chính sau đây:

+ *TCP (Transmission Control Protocol):* Chuyên việc nối các hosts lại và bảo đảm việc giao hàng *(messages)* vì nó vừa dùng sự xác nhận hàng đến *(Acknowledgement)* giống như thư bảo đảm, vừa kiểm xem kiện hàng có bị hư hại không bằng cách dùng CRC *(Cyclic Redundant Check)*, giống như có đóng khằng chỗ mở kiện hàng.

+ *IP (Internet Protocol):* Lo về địa chỉ và chuyển hàng đi đúng hướng, đến nơi, đến chốn.

+ *SMTP (Simple Mail Transfer Protocol):* Chuyên việc giao Email.

+ *FTP (File Transfer Protocol):* Chuyên việc gởi File (upload/download) giữa các hosts.

+ *SNMP (Simple Network Management Protocol):* Dùng cho các programs quản lý mạng để user có thể quản lý mạng từ xa.

+ *UDP (User Datagram Protocol):* Chuyển giao các bọc nhỏ (packets) của một kiện hàng. Nó nhanh hơn TCP ví không có sự kiểm tra hay sửa lỗi. Ngược lại, nó không bảo đảm việc giao hàng.

- Là Network Administrator ta nên làm quen với các công cụ chuẩn để làm việc với TCP/IP như:

+ *File Transfer Protocol (FTP):* Để thử upload/download files giữa các hosts.

+ *Telnet:* Cho ta Terminal Emulation (giả làm một Terminal) để nói chuyện với một Host chạy program Telnet Server.

+ *Packet Internet Groper (Ping):* Dùng để thử TCP/IP configurations và connections.

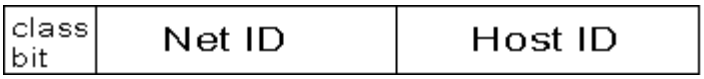
+ *IPCONFIG:* Để kiểm TCP/IP configuration của local host.

+ *NSLOOKUP:* Dùng line command để đọc các records trong DNS (Domain Name System) database.

+ *TRACERT:* Để display các khúc đường (route) dùng giữa hai hosts.

C. Thành Phần và hình dạng của địa chỉ IP

- Địa chỉ IP đang được sử dụng hiện tại (IPv4) có 32 bit chia thành 4 Octet (mỗi Octet có 8 bit, tương đương 1 byte) cách đếm đều từ trái qua phải bít 1 cho đến bít 32, các Octet tách biệt nhau bằng dấu chấm (.), bao gồm có 3 thành phần chính.



Bit 1..... 32

+ Bit nhận dạng lớp (Class bit) để phân biệt địa chỉ ở lớp nào.
+ Địa chỉ của mạng (Net ID)
+Địa chỉ của máy chủ (Host ID).

- Địa chỉ Internet biểu hiện ở dạng bit nhị phân:

x y x y x y x y. x y x y x y x y. x y x y x y x y. x y x y x y x y

(x, y = 0 hoặc 1).

Ví dụ:

0 0 1 0 1 1 0 0. 0 1 1 1 1 0 1 1. 0 1 1 0 1 1 1 0. 1 1 1 0 0 0 0 0

bit nhận dạng Octet 1 Octet 2 Octet 3 Octet 4

- Địa chỉ Internet biểu hiện ở dạng thập phân: **xxx.xxx.xxx.xxx**

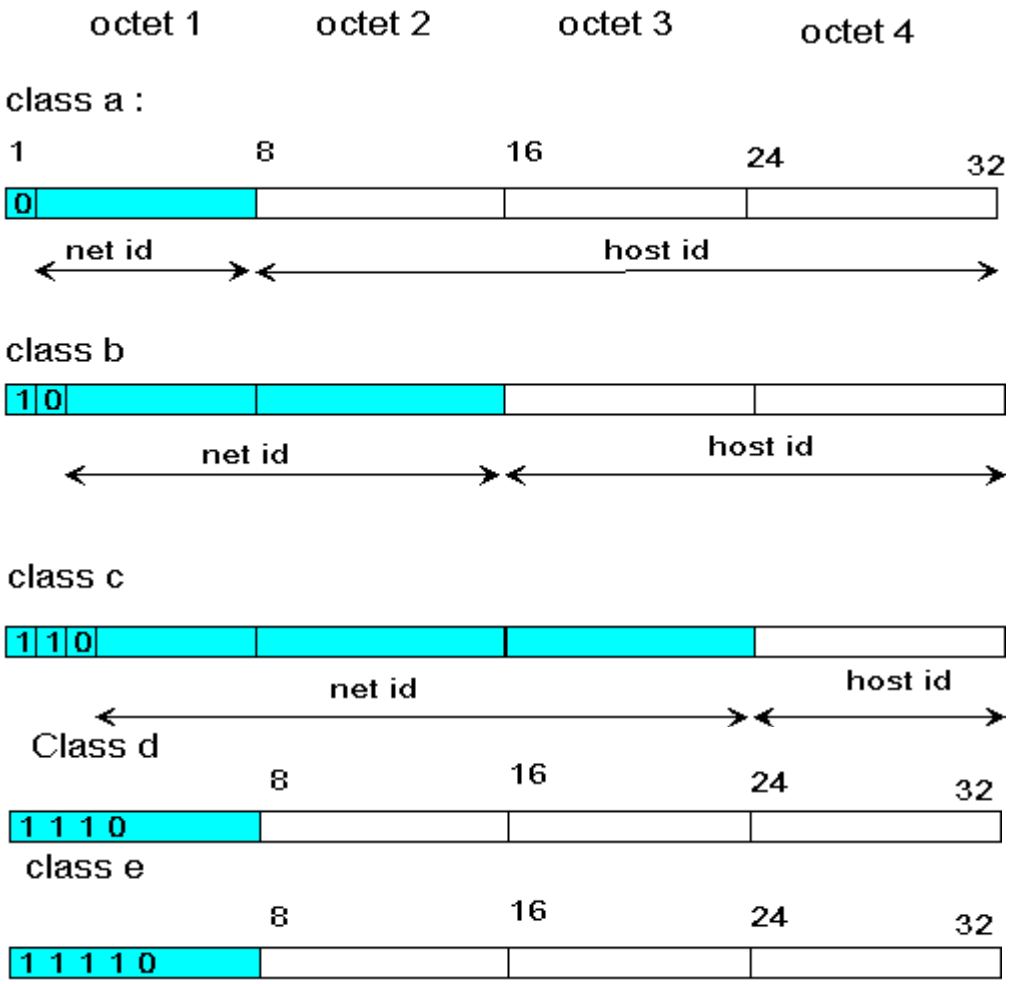
x là số thập phân từ 0 đến 9

Ví dụ: 146. 123. 110. 224

- Dạng viết đầy đủ của địa chỉ IP là 3 con số trong từng Octet. Ví dụ: địa chỉ IP thường thấy trên thực tế có thể là 53.143.10.2 nhưng dạng đầy đủ là 053.143.010.002.

❖ **Các lớp địa chỉ IP**

- Địa chỉ IP chia ra 5 lớp A,B,C, D, E. Hiện tại đã dùng hết lớp A,B và gần hết lớp C, còn lớp D và E Tổ chức internet đang để dành cho mục đích khác không phân, nên chúng ta chỉ nghiên cứu 3 lớp đầu.



Qua cấu trúc các lớp địa chỉ IP chúng ta có nhận xét sau:

♦ Bit nhận dạng là những bit đầu tiên - của lớp A là 0, của lớp B là 10, của lớp C là 110, lớp D có 4 bit đầu tiên để nhận dạng là 1110, còn lớp E có 5 bít đầu tiên để nhận dạng là 11110.

♦ Địa chỉ lớp A: Địa chỉ mạng ít và địa chỉ máy chủ trên từng mạng nhiều cho phép định danh tới 126 mạng(không phân 127) và mỗi mạng có tới 16,777,214 host. Nói cách khác địa chỉ thực thế sẽ từ 001.000.000.001 đến 126.255.255.254, lớp này dùng cho các trạm có số trạm cực lớn.

♦ Địa chỉ lớp B: Địa chỉ mạng vừa phải và địa chỉ máy chủ trên từng mạng vừa phải. Cho phép định danh tới 16,328 mạng và mỗi mạng có đến 65,534 máy chủ, địa chỉ phân trong thực tế se từ 128.001.000.0001 đến 191.254.255.254.

♦ Địa chỉ lớp C: lớp này dùng cho mạng có ít trạm, Địa chỉ lớp C có thể phân cho 2 097 150 mạng và mỗi một mạng có 254 máy chủ. Nói cách khác sẽ từ 192. 000. 001. 001 đến 223. 255. 254.254.

♦ Địa chỉ lớp D: dùng để gởi IP datagram tới một nhóm các host trên mạng, chưa được sử dụng nhiều.

♦ Địa chỉ lớp E: dùng để dự phòng trong tương lai.

D.Subnet Masks

Như đã nêu trên địa chỉ trên Internet thực sự là một tài nguyên, một mạng khi gia nhập Internet được Trung tâm thông tin mạng Internet (NIC) phân cho một số địa chỉ vừa đủ dùng với yêu cầu lúc đó, sau này nếu mạng phát triển thêm lại phải xin NIC thêm, đó là điều không thuận tiện cho các nhà khai thác mạng.

Hơn nữa các lớp địa chỉ của Internet không phải hoàn toàn phù hợp với yêu cầu thực tế, địa chỉ lớp B chẳng hạn, mỗi một địa chỉ mạng có thể cấp cho 65534 máy chủ, Thực tế có mạng nhỏ chỉ có vài chục máy chủ thì sẽ lãng phí rất nhiều địa chỉ còn lại mà không ai dùng được. Để khắc phục vấn đề này và tận dụng tối đa địa chỉ được NIC phân, bắt đầu từ năm 1985 người ta nghĩ đến Địa chỉ mạng con.

Như vậy phân địa chỉ mạng con là mở rộng địa chỉ cho nhiều mạng trên cơ sở một địa chỉ mạng mà NIC phân cho, phù hợp với số lượng thực tế máy chủ có trên từng mạng và Subnet Masks sẽ làm công việc này.

Khi ta chia một Network ra thành nhiều Network nhỏ hơn, các Network nhỏ nầy được gọl là Subnet. Theo quy ước, các địa chỉ IP được chia ra làm 5 Class (lớp) nhưng chỉ có 3 lớp được sử dụng như sau:

Address Class	**Subnet mask trong dạng nhị phân**	**Subnet mask**
Class A	11111111 00000000 00000000 00000000	255.0.0.0
Class B	11111111 11111111 00000000 00000000	255.255.0.0
Class C	11111111 11111111 11111111 00000000	255.255.255.0

Subnet Mask của Class A bằng 255.0.0.0 có nghĩa rằng ta dùng 8 bits, tính từ trái qua phải (các bits được set thành 1), của địa chỉ IP để phân biệt các NetworkID của Class A. Trong khi đó, các bits còn sót lại (trong trường hợp Class A là 24 bits được reset thành 0) được dùng để biểu diễn computers, gọi là HostID.

Subnet Mask là kết hợp của Default Mask với giá trị thập phân cao nhất của các bit lấy từ các Octet của địa chỉ máy chủ sang phần địa chỉ mạng để tạo địa chỉ mạng con.

Subnet Mask bao giờ cũng đi kèm với địa chỉ mạng tiêu chuẩn để cho người đọc biết địa chỉ mạng tiêu chuẩn này dùng cả cho 254 máy chủ hay chia ra thành các mạng con. Mặt khác nó còn giúp Router trong việc định tuyến cuộc gọi.

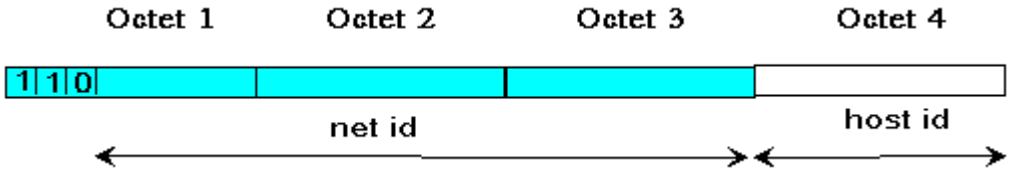
❖ **Nguyên tắc chung:**

♦ Lấy bớt một số bit của phần địa chỉ máy chủ để tạo địa chỉ mạng con.

♦ Lấy đi bao nhiêu bit phụ thuộc vào số mạng con cần thiết (Subnet mask) mà nhà khai thác mạng quyết định sẽ tạo ra.

♦ Vì địa chỉ lớp A và B đều đã hết, hơn nữa hiện tại mạng Internet của Tổng công ty do VDC quản lý đang được phân 8 địa chỉ mạng lớp C nên chúng ta sẽ nghiên cứu kỹ phân chia địa chỉ mạng con ở lớp C.

Ví dụ : *Địa chỉ mạng con của địa chỉ lớp C như sau*



Địa chỉ lớp C có 3 octet cho địa chỉ mạng và 1 octet cuối cho địa chỉ máy chủ vì vậy chỉ có 8 bit lý thuyết để tạo mạng con, thực tế nếu dùng 1 bit để mở mạng con và 7 bit cho địa chỉ máy chủ thì vẫn chỉ là một mạng và ngược lại 7 bit để cho mạng và 1 bit cho địa chỉ máy chủ thì một mạng chỉ được một máy, như vậy không logic, ít nhất phải dùng 2 bit để mở rộng địa chỉ và 2 bit cho địa chỉ máy chủ trên từng mạng. Do vậy trên thực tế chỉ dùng như bảng sau.

Default Mask của lớp C : 255.255.255.0

Địa chỉ máy chủ
<-------->

255.255.255.1 1 0 0 0 0 0 0 ; 192 (2 bit đ/ chỉ mạng con 6 bit đ/chỉ máy chủ)
255.255.255.1 1 1 0 0 0 0 0 ; 224 (3 bit đ/chỉ mạng con 5 bit đ/chỉ máy chủ)
255.255.255.1 1 1 1 0 0 0 0 ; 240 (4 bit đ/chỉ mạng con 4 bit đ/chỉ máy chủ)
255.255.255.1 1 1 1 1 0 0 0 ; 248 (5 bit đ/chỉ mạng con 3 bit đ/chỉ máy chủ)
255.255.255.1 1 1 1 1 1 0 0 ; 252 (6 bit đ/chỉ mạng con 2 bit đ/chỉ máy chủ)

<----------> <--------->
Default Mask *Địa chỉ mạng con*

Trường Hợp	*Subnetmask*	Số Lượng mạng con	Số máy chủ trên từng mạng
1	255.255.255.192	2	62
2	255.255.255.224	6	30
3	255.255.255.240	14	14
4	255.255.255.248	30	6
5	255.255.255.252	62	2

Như vậy một địa chỉ mạng ở lớp C chỉ có 5 trường hợp lựa chọn trên (Hay 5 Subnet Mask khác nhau), tuỳ từng trường hợp cụ thể để quyết định số mạng con

Tương tự cách phân chia mạng con của lớp A, B cũng như cách phân chia của lớp C.

CHƯƠNG 3

CÁC GIAO THỨC TRUYỀN NHẬN MAIL

I Các khái niệm cơ bản

Các hệ thống thư điện tử thường bao gồm hai hệ thống con: các tác nhân *người sử dụng (the user agents - gọi tắt là UA), nó cho phép chúng ta* đọc và gửi thư, và các tác nhân truyền thông điệp (the message transfer agents - gọi tắt là MTA), nó làm nhiệm vụ chuyển các thông điệp từ nguồn đến đích. Các UAs là các chương trình cục bộ hỗ trợ dựa trên điều khiển bằng lệnh, trình đơn menu hay dùng phương pháp đồ hoạ để tương tác với hệ thống thư điện tử. Các MTAs là các trình tiện ích hoạt động ở chế độ nền (background) thực hiện các nhiệm vụ cần thiết như tiếp nhận thư điện tử và chuyển thư qua các hệ thống. Đặc biệt, các hệ thống thư điện tử hỗ trợ năm chức năng cơ bản, được mô tả dưới đây:

- Composition: Xử lý việc tạo các thông điệp và trả lời. Cho phép bất cứ trình soạn thảo nào có thể được sử dụng cho phần thân của thông điệp, các hệ thống có thể tự nó đảm trách việc đánh địa chỉ và chỉ số các trường tiêu đề (header fields) được kèm theo cùng với mỗi thông điệp. Ví dụ như, khi trả lời một thông điệp, hệ thống thư điện tử có thể tách địa chỉ của người gửi từ các thư được gửi đến và tự động chèn nó vào các trường thích hợp trong phần hồi âm (reply).

- Transfer: Làm nhiệm vụ chuyển các thông điệp từ người gửi đến nơi người nhận. Trong phần này, việc chuyển các thông điệp yêu cầu phải thiết lập một kết nối đến đích (người nhận) hay một số thao tác của thiết bị như xuất thông điệp và kết thúc việc kết nối. Hệ thống thư điện tử làm việc này một cách tự động mà không cần có một sự can thiệp nào của người sử dụng.

- Reporting: Buộc phải thực hiện để báo cho người gửi những gì xảy ra đối với thông điệp vừa gửi là ở tình huống đã gửi đến đích chưa? hoặc việc gửi đã bị huỷ bỏ? hoặc thư đã bị lạc?.

- Displaying: Những thông điệp gửi đến được yêu cầu làm sao để mọi người có thể đọc được thư của họ. Đôi khi người ta yêu cầu quá trình chuyển đổi hay một trình hiển thị đặc biệt để hỗ trợ, ví dụ như, nếu thông điệp có dạng một tệp PostScript hay tiếng nói được số hoá kèm theo trong thông điệp gửi đến.

- Disposition: Là bước cuối cùng liên quan đến những gì người nhận thực hiện đối với thông điệp sau khi đã nhận nó. Những khả năng có thể là ném nó đi trước khi đọc, ném nó đi sau khi đọc, lưu nó, v ..v. Nó cũng sẽ có thể thu nhận để đọc lại với các thông điệp đã được lưu lại, chuyển tiếp chúng hoặc xử lý chúng bằng những phương pháp khác nhau khi được yêu cầu của người sử dụng.

Thêm vào đó các dịch vụ này, hầu hết các hệ thống thư điện tử cung cấp nhiều đặc tính nâng cao khác nhau. Một số đặc tính tiêu biểu như, khi người ta muốn chuyển thư hay khi họ nghĩ xa hơn về các chi tiết về thời gian, có lẽ họ muốn thư của họ được chuyển tiếp, chính vì thế mà hệ thống thực hiện điều này một cách tự động.

Hầu hết các hệ thống cho phép người sử dụng tạo các hộp thư (mailboxes) để lưu trữ các thư chuyển đến (incoming email). Các lệnh được người ta yêu cầu tạo và huỷ bỏ các hộp thư, kiểm tra các nội dung hộp thư, chèn và xoá các thông điệp khỏi hộp thư, v..v.

Những người giám đốc công ty thường cần gửi một thông điệp đến mỗi người trong số những người cấp dưới, những khách hàng, hay đến các nhà cung cấp. Thì điều này đưa ra một ý tưởng về danh sách thư (mailing list), nó là một danh sách các địa chỉ thư điện tử. Khi một thông điệp được gửi đến *mailing list,* các bản sao giống hệt được phát đến mọi người có địa chỉ trên danh sách. Một ý tưởng quan trọng khác là thư điện tử được đăng ký, để cho phép người gửi (sender or originator) biết thư của họ đã đến. Việc thông báo tự động của các thư không được phát đi một cách luân phiên để người ta có thể biết. Trong bất kỳ trường hợp nào, người gửi nên có một số điều khiển thông qua thông báo những gì xảy ra.

1. Cấu trúc của một bức thư:

Về cơ bản, một bức Mail bao gồm 3 phần chính:

- *Phần phong bì:* Mô tả thông tin về người gởi và người nhận. Do hệ thống tạo ra.
- *Phần tiêu đề (header):* chứa đựng các thông tin về người gởi, người nhận, chủ đề bức Mail, địa chỉ hồi âm .v.v.. Các thông tin này, một số được người sử dụng cung cấp khi gởi Mail, một số khác được chương trình Mail thêm vào, và số còn lại do Hệ thống điền thêm.
- *Phần nội dung (body):* chứa đựng nội dung của bức Mail, là nội dung được tạo ra bởi trình soạn thảo Editor của chương trình Mail. Sau đây là chi tiết của từng phần:

a. Phần phong bì (Envelope):

Phần này do các MTA tạo ra và sử dụng, nó chứa các thông tin để chuyển nhận email như địa chỉ của nơi nhận, địa chỉ của nơi gửi. Hay nói cách khác, giao thức SMTP sẽ quy định thông tin của phong bì, các hệ thống Email cần những thông tin này để chuyển dữ liệu từ một máy tính này sang một máy tính khác.

b. Phần tiêu đề (header):

- Phần này cung cấp những thông tin tổng quát về Email như người nhận, người gửi, ngày giờ nhận...
- Cấu tạo gồm nhiều trường (field) cấu trúc mỗi trường là một dòng văn bản ASCII chuẩn 7 bit như sau: <tên trường >: <nội dung của trường>.
- Sau đây là một số trường thông dụng và ý nghĩa của nó :
  - ✓ Date: chỉ ngày giờ nhận mail.
  - ✓ From: chỉ người gởi.
  - ✓ To: chỉ người nhận.
  - ✓ Cc: chỉ người những nhận bản copy của mail.
  - ✓ Bcc: chỉ ra những người nhận bản copy của bức mail, nhưng từng người không biết những người nào sẽ nhận bức thư này
  - ✓ Return-path: chứa các thông tin để người nhận có thể trả lời lại (thường nó chính là địa chỉ người gởi).

✓ Subject: chủ đề của nội dung Email.

Các trường trên là các trường chuẩn do giao thức SMTP quy định, ngoài ra trong phần header cũng có thể có thêm một số trường khác do chương trình Email tạo ra nhằm quản lý các email mà chúng tạo. Các trường này được bắt đầu bằng ký tự X- và thông tin theo sau là cũng giống như ta thấy trên một trường chuẩn.

c. Phần nội dung (body):

Để phân biệt phần tiêu đề và phần nội dung của bức Mail, người ta qui ước đặt ranh giới là một dòng trắng (chuỗi ký tự "\r\n"). Kết thúc của phần nội dung là chuỗi ký tự kết thúc Mail: "\r\n.\r\n". Như vậy nội dung bức Mail nằm trong khoảng giữa dòng trắng đầu tiên và ký tự kết thúc Mail, và trong phần nội dung của bức Mail không được phép tồn tại chuỗi ký tự kết thúc Mail. Mặt khác do môi trường truyền thông là mạng Internet nên các ký tự cấu thành phần body của bức Mail cũng phải là các ký tự ASCII chuẩn.

2. Tác nhân người sử dụng (The User Agent)

Các hệ thống thư điện tử có hai phần cơ bản, như chúng ta đã thấy gồm: phần UA và phần MTA. Trong phần này chúng ta sẽ xét đến phần UA. Một *UA* thường là một chương trình (đôi khi được gọi là bộ phận đọc thư) nó nhận một trong những lệnh khác nhau như là cho mục đích soạn thư, nhận thư, và hồi đáp các thông điệp, cũng như việc thao tác trên các hộp thư (mailboxes). Một số UA (User Agent) có giao diện trình đơn (menu) hay biểu tượng (icon) khá hấp dẫn mà nó yêu cầu sử dụng chuột hoặc chấp nhận các lệnh 1 ký tự từ bàn phím có cùng chức năng với menu và các icon.

3. Gửi thư (Sending Email)

Để gửi đi một thông điệp, người sử dụng phải cung cấp thông điệp, địa chỉ đích và một số tham số khác nếu có (ví dụ như là mức ưu tiên hay bảo mật). Người sử dụng có thể tạo thông điệp với một trình soạn thảo văn bản khác nhau, một chương trình xử lý từ hay với bộ soạn thảo được xây dựng trên UA. Địa chỉ đích phải có một định dạng mà làm sao cho UA có thể hiểu được. Nhiều UA tiếp nhận các địa chỉ DNS (Domain Name System) có dạng mailbox@location.

4. Đọc thư (Reading Email)

Khi UA được khởi động nó kiểm tra xem trong hộp thư của người sử dụng có thư gửi đến không trước khi hiển thị các thứ khác lên màn hình. Khi đó có lẽ nó sẽ thông báo một số các thông điệp trong hộp thư hay hiển thị một dòng vắn tắt của mỗi thông điệp và chờ nhận lệnh để xử lý. Một ví dụ ở hình 1.8 cho thấy một viễn cảnh sau khi UA khởi động hiển thị những yêu cầu vắn tắt của các thông điệp. Trong ví dụ này hộp thư (mailbox) gồm có tám thông điệp.

Mỗi dòng hiển thị chứa một số trường được trích ra từ phong thư hay phần đầu (header) của từng thông điệp được định vị trong hộp thư. Trong một hệ thống thư điện tử đơn giản, sự lựa chọn của các trường hiển thị được người ta xây dựng thành một chương trình. Trong các hệ thống phức tạp hơn, người sử dụng có thể xác định cho các trường nào được hiển thị bằng cách

cung cấp một hiện trạng người sử dụng (User Profile), hay một tệp mô tả định dạng hiển thị. Trong ví dụ này, trường đầu tiên là số thông điệp có trong hộp thư. Trường thứ hai, là các cờ có thể chứa một kí tự K, có nghĩa là thông điệp cũ đã được đọc kỳ trước rồi và được lưu lại trong hộp thư; kí tự A có nghĩa là thư này đã được hồi âm rồi; ký tự F (có thể có), có nghĩa là thư này được chuyển tiếp đến người khác. Các cờ khác nữa cũng có thể được đưa vào ngoài những cờ này.

#	Flags	Bytes	Sender	Subject
1	K	1030	Asw	Changes to MINIX
2	KA	6348	Radia	Comments on material you sent me
3	KF	4519	Amy N. Wong	Request for information
4		1236	Bal	Deadline for grant proposal
5		103610	Kaashoek	Text of DCS paper
6		1223	Emily E.	Pointer to WWW page
7		3110	Saniya	Referee reports for the page
8		1204	Dmr	Re: My student's visit

Hình 3.1 Hiển thị các nội dung của hộp thư.

Trường thứ ba cho biết chiều dài của thông điệp và trường thứ tư cho biết ai là người gửi thông điệp. Vì trường này được trích ra từ các thông điệp rất đơn giản nên trường này có thể chứa các tên, họ tên đầy đủ, các tên viết tắt, các tên đăng nhập, hay bất cứ thứ gì mà người gửi có thể đặt vào trong trường này. Cuối cùng là trường chủ đề thư (Subject) cho biết một câu vắn tắt về những gì trong nội dung thông điệp. Những người nào quên điền vào trường này thì thường được cho là những câu trả lời cho thư của họ là không chú ý đến mức ưu tiên cao nhất.

Sau khi các phần đầu đã được hiển thị, người sử dụng có thể thực hiện bất cứ lệnh nào có thể. Một chọn lựa tiêu biểu được liệt kê ở bảng bên dưới (hình 1.9) là một ví dụ khi một người sử dụng bằng hệ thống Mmdf của hệ điều hành UNIX. Có một số lệnh yêu cầu có tham số. Ký hiệu # có nghĩa là chỉ số của một thông điệp (hay có thể có nhiều thông điệp) được chấp nhận. Tương tự, mẫu tự a có thể được sử dụng có nghĩa cho tất cả các thông điệp.

5. Định dạng thông điệp (Message Formats)

Chúng ta bây giờ hãy quay đến từ giao diện người sử dụng đến định dạng của các thông điệp thư điện tử. Trước tiên chúng ta xét thư điện tử dựa trên bản mã ASCII sử dụng chuẩn RFC 822 (Request for Comments). Sau đó xét đến các mở rộng đa phương tiện cho chuẩn RFC 822.

II.Chuẩn RFC 822

- Các thông điệp bao gồm một phong bì gốc (được mô tả trong chuẩn RFC 821), một số các trường cho phần đầu (header), một dòng để trống và sau đó

là phần thân (body). Mỗi trường header bao gồm các dòng văn bản ASCII chứa tên trường, dấu hai chấm, và cho hầu hết các trường đều có một giá trị. RFC 822 là một chuẩn cũ và giữa các trường header của phong bì (envelope) không phân biệt rõ ràng như một chuẩn mới khác. Khi sử dụng, thông thường UA xây dựng một thông điệp và đưa nó qua bộ phận tác nhân truyền thông *điệp (message transfer agents - MTA), ở đây nó dùng một số các trường* header để xây dựng một envelope thực sự, thông điệp được thay đổi bởi cái cũ đi một chút cùng với envelope.

Command	**Parameter**	**Description**
H	#	Display header(s) on the screen
C		Display current header only
T	#	Type message(s) on the screen
S	Address	Send a message
F	#	Forward message(s)
A	#	Answer message(s)
D	#	Delete message(s)
U	#	Undelete previously deleted message(s)
M	#	Move message(s) to another mailbox
K	#	Keep message(s) after exiting
R	Mailbox	Read a new mailbox
N		Go to the next message and display it
B		Backup to the previous message and display it
G	#	Go to a specific message but do not display it
E		Exit the mail system and update the mailbox

Hình 3.2: Các lệnh điều khiển thư đặc biệt

- Các trường header chủ yếu liên quan đến việc chuyển giao thông điệp được liệt kê dưới bảng sau. Trường To: trường này cho biết địa chỉ DNS của người nhận đầu tiên. Trường hợp nhiều người nhận cũng có thể cho phép. Trường Cc: cho biết địa chỉ của những người nhận kế tiếp (còn gọi là địa chỉ đồng gửi). Trong các thuật ngữ của việc phát thư, không có sự phân biệt giữa những người nhận thứ nhất và người nhận thứ hai. Thuật ngữ Cc (Carbon copy) là một mẫu đã được xác định, vì máy tính không sử dụng các trang giấy bản sao. Trường Bcc: (Blind carbon copy) giống như trường Cc: chỉ trừ là dòng này được xoá khỏi tất cả các bản sao được gửi đến những người nhận đầu tiên và người nhận thứ hai. Đặc tính này cho phép người ta gửi các bản sao đến những người trong nhóm thứ ba mà trong đó không có người thứ nhất và người thứ hai biết.

Header	***Meaning***
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Hình 3.3 Các trường header RFC 822 liên quan trong việc truyền thông điệp

- Hai trường kế tiếp, From và Sender cho biết để phân biệt người viết và người gửi thông điệp. Hai trường này hoàn toàn không giống nhau. Ví dụ một nhà quản trị doanh nghiệp có thể viết một thông điệp nhưng cô thư ký là người thật sự truyền nó đi. Trong trường hợp này, người quản trị phải được liệt kê vào trong trường From: và cô thư ký trong trường Sender:.

Header	**Meaning**
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message latter
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User chosen keywords
Subject:	Short summary of the message for the one-line display

Hình 3.4 : Một số trường được sử dụng trong header thông điệp RFC 822.

- Trường From: yêu cầu phải có còn trường Sender: có thể được bỏ qua nếu việc viết và gửi cùng một người. Các trường này cần thiết khi trong trường hợp thông điệp không được phát đi và phải được trả lại cho người gửi. Dòng chứa trường Received được đưa vào bởi các MTAs dọc theo đường truyền. Dòng này chứa định danh của agent, ngày tháng và thời gian thông điệp được nhận, và các thông tin khác có thể được sử dụng cho việc tìm kiếm các lỗi trong hệ thống định tuyến.

- Trường Return-Path: được đưa vào bởi MTAs cuối cùng và được dùng cho việc gửi trở lại người gửi. Theo lý thuyết, thông tin này có thể được tập hợp lại từ các header Received: (loại trừ tên của hộp thư người gửi), nhưng nó ít khi được điền đầy đủ như thế và chỉ đặc biệt chứa địa chỉ của người gửi.

❖ **MIME** (Multipurpose Internet Mail Extension)

Một giao thức Internet mới mẻ được phát triển để cho phép trao đổi các thông điệp thư điện tử có nội dung phong phú thông qua mạng không đồng nhất (heterogeneous network), máy móc, và các môi trường thư điện tử. Trong thực tế, MIME cũng đã được sử dụng và mở rộng bởi các ứng dụng không phải thư điện tử. Hiện nay, trên mạng diện rộng Internet, đối với RFC 822 chỉ làm những công việc định nghĩa các header nhưng còn nội dung bên trong thì vẫn còn lỗi thời, chính vì thế mà vấn đề này không còn thích hợp nữa. Các vấn đề bao gồm việc gửi và nhận thư như sau:

- ✓ Những thông điệp sử dụng các ngôn ngữ có dấu.
  ví dụ: Tiếng Pháp và tiếng Đức.
- ✓ Những thông điệp sử dụng các ngôn ngữ không phải chữ cái Latin.
  ví dụ: Tiếng Do thái, tiếng Nga. . .
- ✓ Những thông điệp sử dụng các ngôn ngữ không có trong các bảng chữ cái.
  ví dụ: Tiếng Trung Quốc, tiếng Nhật. . .
- ✓ Những thông điệp sử không chứa văn bản.
  ví dụ: Có âm thanh và hình ảnh.

- Một giải pháp đã được đưa ra trong RFC 1341 và được cập nhật mới nhất trong RFC 1521. Giải pháp này được gọi là MIME, hiện nay được sử dụng rộng rãi.

Khái niệm cơ bản của MIME là tiếp tục sử dụng định dạng RFC 822, nhưng thêm cấu trúc vào phần thân của thông điệp và định nghĩa các nguyên tắc mã hóa các thông điệp không phải các bảng mã ASCII. Để khỏi bị lệch hướng của RFC 822, các thông điệp MIME có thể được gửi đi được sử dụng các giao thức và chương trình thư hiện có. Tất cả các chương trình này phải được thay đổi thành các chương trình gửi và nhận sao cho người dùng có thể dùng được.

MIME định nghĩa năm header thông điệp mới được trình bày trong hình 1.12. Các header này trước tiên báo cho UA nhận thông điệp mà nó đang dùng bằng thông điệp MIME và phiên bản của MIME đang dùng. Bất cứ thông điệp nào không chứa header MIME-Version: được giả định là một thông điệp hình thức được mã hóa bằng tiếng Anh và nó được xử lý như thế.

Header	**Meaning**
MIME-Version:	Indentifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Nature of the message

Hình 3.6 Các header RFC 822 được MIME thêm vào.

- Bảy kiểu chính mô tả MIME được định nghĩa trong RFC 1521, mỗi kiểu của nó lại có một hay nhiều kiểu phụ. Kiểu chính và kiểu phụ (xem hình 3.6) được phân biệt bởi một dấu vạch chéo, như có dạng sau: Content-Type: *video/mpeg*

Type	**Subtype**	***Description***
Text	Plain	Unformatted text
	Richtext	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Applicatio n	Octel-stream	An uninterpreted byte sequence
	Postscript	A printable document in Postscript
Message	Rfc 822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 message

Hình 3.7 Các kiểu chính và kiểu phụ được định nghĩa trong RFC 1521

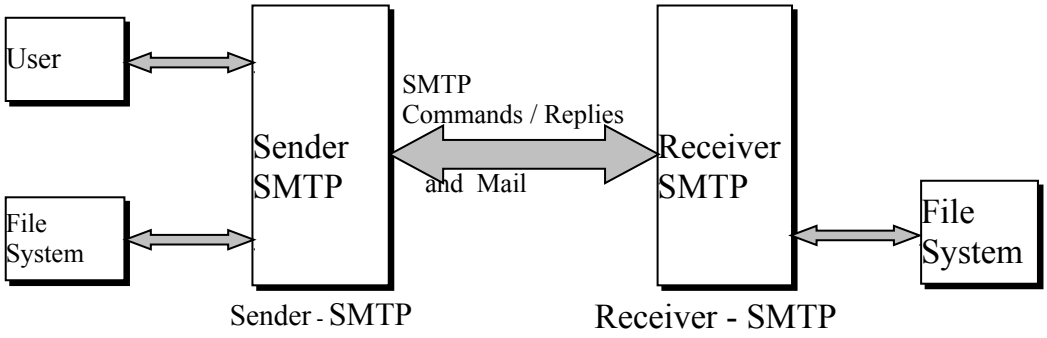
❖ **Truyền thông điệp (Message Transfer)**

Hệ thống truyền thông điệp có liên quan tới việc chuyển tiếp (relaying) các thông điệp từ người gửi đến người nhận. Phương pháp đơn giản nhất để thực hiện điều này là thiết lập một kết nối truyền thông từ máy nguồn đến máy đích lúc đó mới truyền các thông điệp đi. Sau khi xem xét nó thực hiện như thế nào, chúng ta sẽ xét một số tình huấn mà ở đó nó không thực hiện và chúng có thể thực hiện về những gì.

III.GIAO THỨC SMTP(RFC821)

- Mục đích của giao thức SMTP là truyền mail một cách tin cậy và hiệu quả. Giao thức SMTP không phụ thuộc vào bất kỳ hệ thống đặc biệt nào và nó chỉ yêu cầu trật tự của dữ liệu truyền trên kênh truyền đảm bảo tính tin cậy.
- Giao thức SMTP được thiết kế dựa vào mô hình giao tiếp sau: khi có yêu cầu từ user về dịch vụ mail, sender-SMTP thiết lập một kênh truyền hai chiều tới reciever-SMTP. Reciever- SMTP có thể là đích cuối cùng hoặc chỉ là đích

trung gian nhận mail. Các lệnh trong giao thức SMTP được sender-SMTP gởi tới reciever-SMTP và reciever-SMTP gởi đáp ứng trở lại cho sender-SMTP.



Hình 3.8 Mô hình tổng quát sử dụng giao thức SMTP

1. *Ý nghĩa các lệnh của một phiên giao dịch SMTP Server:*

- Những lệnh SMTP định nghĩa sự truyền mail hay chức năng của hệ thống mail được yêu cầu bởi user. Những lệnh SMTP là những chuỗi ký tự kết thúc bằng <CRLF>. Bản thân mã lệnh là những ký tự chữ (alphabetic) kết thúc bởi <SP> nếu có những tham số theo sau và nếu không có thì <CRLF>. Cú pháp của những mailbox phải tuân theo những qui ước của receiver.

- Một phiên giao dịch mail chứa đựng một vài đối tượng dữ liệu, được truyền như là những đối số cho các lệnh khác nhau. Reverse-path là đối số của lệnh MAIL. Forward-path là đối số của lệnh RCPT. Và mail data là đối số của lệnh DATA. Những đối số hay những đối tượng dữ liệu này được truyền đi và duy trì cho đến khi xác nhận truyền xong bởi sự chỉ định kết thúc của mail data. Mô hình hiện thực cho cách làm này là những buffer riêng biệt được cung cấp để lưu trữ kiểu của đối tượng dữ liệu, đó là các buffer reverse-path, forward-path, và mail data buffer. Những lệnh xác định tạo ra thông tin được gắn vào một buffer xác dịnh, hoặc xoá đi một hay một số buffer nào đó.

♦ **HELLO (HELO)**

Lệnh này được dùng để xác định ra ai là người gởi mail. Vùng đối số chứa host name của bên gởi.

Bên nhận định danh cho nó đối với sender thông qua việc bắt tay trả lời kết nối.

Với lệnh này và sự trả lời OK để xác định rằng cả sender và reciever đang ở trạng thái khởi đầu, tất cả các bảng trạng thái và buffer đã được xoá sạch.

♦ **MAIL**

Lệnh này được dùng để khởi tạo quá trình trao đổi mail mà ở đó mail data được phân phát tới một hay nhiều mailbox. Vùng đối số của lệnh có chứa reverse-path.

Reverse-pat bao gồm một danh sách tuỳ ý các host và mailbx của sender. Khi danh sách của host được chỉ ra, nó là lộ trình nguồn trở về (reverse source route) và chỉ ra các host mà mail sẽ được truyền tiếp vận qua các host trong danh sách đó. Danh sách này được sử dụng như là một lộ trình nguồn để trả lời thông báo không phân phát được cho sender. Mỗi khi truyền tiếp vận host sẽ thêm vào phần định danh của nó vào đầu danh sách, nó phải sử dụng tên của nó khi đã được biết trong IPCE nơi mà nó đang truyền tiếp vận mail hơn là IPCE mà mail đã tới(nếu chúng khác nhau).

Lệnh này sẽ xoá các buffer sau: reverse-path, forward-path, và mail data buffer, và nó thêm thông tin của reverse-path từ lệnh này vào reverse-path buffer.

♦ **RECIPIENT (RCPT)**

Lệnh này được sử dụng để định ra một người nhận mail, nhiều người nhận (cùng một nội dung mail) sẽ được xác định bằng cách gởi nhiều lệnh này.

Forward - path bao gồm một danh sách tuỳ ý các host và một hộp thư đích cần thiết. Khi danh sách này được chỉ ra, đó là lộ trình nguồn và cho biết mail sẽ được truyền tiếp vận tới host kế tiếp nằm trong danh sách. Nếu reciever-SMTP không được hiện thực chức năng truyền tiếp vận thì thông báo trả về có thể là : không biết local user (550).

Khi mail đã được truyền tiếp vận, host làm công việc này phải bỏ phần định danh nó từ chỗ bắt đầu forward-path và đặt nó vào chỗ bắt đầu của reverse-path. Khi mail đến được đích cuối cùng rồi, reciever-SMTP bỏ nó vào trong mailbox với sự đồng ý của host mail đó.

Lệnh này sẽ chèn đối số là forward-path vào forward-path buffer.

♦ **DATA**

Reciever sẽ xử lý những dòng theo sau lệnh khi mail data đến từ sender. Lệnh này tạo ra mail data để đặt vào mail data buffer. Mail data có thể chứa bất kỳ ký tự nào trong bộ mã ASCII.

Mail data được kết thúc bởi một dòng mà nó chỉ chứa một dấu chấm “ .”.

Sự kết thúc mail data để yêu cầu receiver phải xử lý việc lưu trữ thông tin trong phiên giao dịch mail ngay. Quá trình xử lý này sử dụng thông tin nằm trong reverse-path buffer, trong forward-path buffer, và trong mail data buffer, khi hoàn tất lệnh này những buffer này sẽ bị xoá. Nếu quá trình xử lý thành công, reciever phải gởi trả lời OK. Nếu bị lỗi, reciever phải gởi thông báo lỗi.

Khi reciever chấp nhận một message cho sự truyền tiếp vậnän hoặc phân phát đến đích cuối cùng, nó thêm vào chỗ khởi đầu của mail data một dòng đánh dấu thời gian. Dòng đánh dấu thời gian chỉ ra định danh của host mà nó nhận message, và ngày tháng và thời gian mà mailđược nhận. Những message được truyền tiếp vận sẽ có nhiều dòng đánh dấu thời gian.

Khi reciever tạo ra “final delivery” của một message, nó thêm vào đầu của mail data một dòng đường dẫn quay về. Đường dẫn quay về duy trì thông tin trong <reverse-path> từ lệnh MAIL. Ở đây, “final delivere” có nghĩa là message thoát khỏi môi trường SMTP. Thông thường điều này có nghĩa là nó

đã được phân phát tới user đích, nhưng trong một vài trường hợp nó có thể được xử lý tiếp và được truyền đi bằng một hệ thống mail khác.

Có thể đối với mailbox, đường dẫn quay về có thể khác với mailbox thực sự của người gởi, ví dụ như có thông báo lỗi đặc biệt được truyền đi để điều khiển mailbox.

♦ **SEND**

Lệnh này được dùng để khởi tạo sự truyền mail mà ở đó maildata sẽ được truyền đi tới một hay nhiều terminal. Vùng đối số chứa phần reverse-path. lệnh thực thi thành công khi message được phân phát tới terminal.

Reverse-path bao gồm một danh sách tuỳ ý các host và mailbox của sender. Khi danh sách của host được chỉ ra, nó là lộ trình nguồn quay về và chỉ ra rằng mail đã được truyền tiếp vận thông qua mỗi host trên danh sách. Danh sách này được dùng như là lộ trình nguồn để trả về thông báo non-delivery cho sender. Mỗi khi truyền tiếp vận, host thêm phần định danh của chính nó vào chỗ bắt đầu của danh sách, nó phải sử dụng tên của nó khi đã biết trong IPCE mà ở đó mail được truyền tiếp vận hơn là mail được truyền tới (nếu chúng có sự khác nhau).

Lệnh nay sẽ xoá các buffer sau : reverse-path, forward-path, và mail data buffer, đồng thời nó thêm reverse-path ở lệnh này vào reverse-path buffer.

♦ **SEND OR MAIL (SOML)**

Lệnh này được sử dụng để khởi tạo sự truyền mail mà ở đó mail data một hay nhiều terminal hoặc các mailbox. Đối với người nhận, mail data được phân phát tới terminal của người nhận nếu người nhận có tích cực, trái lại, là mailbox của người nhận. Lệnh này thành công khi message được phân phát tới terminal hoặc là mailbox.

Reverse-path bao gồm một danh sách tuỳ ý các host và mailbox của sender. Khi danh sách này được chỉ ra,nó là lộ trình nguồn quay về và chỉ ra mail đã được truyền tiếp vận thông qua những host trong danh sách. Danh sách này được dùng như là lộ trình nguồn để trả về thông báo non-delivery cho sender. Mỗi khi có sự truyền tiếp vận, host thêm phần định danh của chính nó vào đầu danh sách, nó phải sử dụng tên của nó khi đã biết trong IPCE mà ở đó mail được truyền tiếp vận hơn là mail được truyền tới (nếu chúng có sự khác nhau).

Lệnh này sẽ xoá đi các buffer sau: reverse-path, forward-path, và mail data buffer, đồng thời nó thêm thông tin reverse-path từ lệnh này vào reverse-path buffer.

- **SEND AND MAIL (SAML)**

Lệnh này được sử dụng để khởi tạo sự truyền mail mà ở đó mail data một hay nhiều terminal hoặc các mailbox. Đối với người nhận, mail data được phân phát tới terminal của người nhận nếu người nhận có tích cực, và đối với mọi người nhận mail sẽ tới mailbox của những người nhận đó.

Vùng đối số chứa đựng một reverse-path. Lệnh này thành công khi, message được phân phát tới mailbox.

Reverse-path bao gồm một danh sách tuỳ ý các host và mailbox của sender. Khi danh sách này được chỉ ra,nó là lộ trình nguồn quay về và chỉ ra mail đã được truyền tiếp vận thông qua những host trong danh sách. Danh sách này được dùng như là lộ trình nguồn để trả về thông báo non-delivery

cho sender. Mỗi khi có sự truyền tiếp vận, host thêm phần định danh của chính nó vào đầu danh sách, nó phải sử dụng tên của nó khi đã biết trong IPCE mà ở đó mail được truyền tiếp vận hơn là mail được truyền tới (nếu chúng có sự khác nhau).

Lệnh này sẽ xoá đi các buffer sau: reverse-path, forward-path, và mail data buffer, đồng thời nó thêm thông tin reverse-path từ lệnh này vào reverse-path buffer.

♦ **RESET (RSET)**

Lệnh này xác định sự truyền mail hiện tại đã bị huỷ bỏ. Các sender, recipient, mail data đã lưu sẽ bị huỷ bỏ và tất cả các bảng trạng thái, các buffer bị xoá. Receiver phải gửi một reply OK.

♦ **VERIFY (VRFY)**

Lệnh này yêu cầu receiver xác nhận đối số là định danh một user. Nếu nó là một user name, full name của user đó (nếu receiver biết) và mailbox đặc tả đầy đủ được trả về.Lệnh này không ảnh hưởng đến reverse-path buffer, forward-path buffer và data mail buffer.

♦ **EXPAND (EXPN)**

Lệnh này yêu cầu receiver xác nhận đối số là một mailing list(danh sách địa chỉ) và trả về một thành phần trong danh sách đó. Full name của các user (nếu biết) và những mailbox đã xác định đầy đủ được trả về trong một reply gồm nhiều dòng.Lệnh này không ảnh hưởng đến reverse-path buffer, forward-path buffer và data mail buffer.

♦ **HELP**

Lệnh này cho receiver những thông tin giúp đỡ cho sender. Lệnh này có thể nhận một đối số (có thể là tên lệnh) và trả về thông tin chi tiết. Lệnh này không ảnh hưởng đến reverse-path buffer, forward-path buffer và data mail buffer.

♦ **NOOP**

Lệnh này không ảnh hưởng các tham số hay các lệnh được đưa vào trước nó, nó đặc tả không có một hành động nào khác hơn là receiver gửi một reply OK. Lệnh này không ảnh hưởng đến reverse-path buffer, forward-path buffer và data mail buffer.

♦ **QUIT**

Lệnh này định rõ receiver phải gửi một reply OK và sau đó đóng kênh truyền. Receiver sẽ không đóng kênh truyền cho đến khi nó nhận và trả lời cho lệnh QUIT (ngay cả nếu có một lỗi xảy ra). Sender sẽ không đóng kênh truyền cho đến khi nó gửi một lệnh QUIT và nhận reply đó (ngay cả nếu có một lỗi trả lời cho lệnh trước đó). Nếu mà kết nối bị đóng trước thời gian mong muốn receiver sẽ làm việc như nếu vừa nhận được một lệnh RSET (bỏ tất cả các giao dịch đang treo mà chưa làm, nhưng không “undo” những đã truyền hoàn tất trước đó) sender sẽ hành động ngay khi lệnh hay quá trình truyền đó trong quy trình nhận được một lỗi tạm thời (4xx).

♦ **TURN**

Lệnh này xác định receiver phải gửi một trong hai reply sau: (1) reply OK và sau đó nhận vai trò của một sender-SMTP, hay (2) gửi một reply từ chối và giữ lại vai trò một receiver-SMTP.

Nếu program-A hiện tại là một sender-SMTP và nó gửi một lệnh TURN và nhận một reply OK (250) thì program-A trở thành receiver-SMTP sau đó program-A sẽ trong trạng thái khởi động ngay khi kênh truyền đã được mở, và sau đó nó gởi lời chào là hỏi dịch vụ đã sẵn sàng (220). Nếu chương trình B hiện tại là reciever và nó nhận được lệnh TURN và nó trả lời OK thì B trở thành sender. B khi đó ở trạng thái khởi tạo ngay khi kênh truyền được mở, và nó chờ nhận trả lời dịch vụ đã sẵn sàng (220).

Để từ chối thay đổi vai trò receiver gửi một reply 502.

Có một vài hạn chế về trật tự khi dùng những lệnh này.Đầu tiên trong một phiên trao đổi phải là lệnh HELLO, lệnh này có thể được dùng sau đó trong một cuộc trao đổi khác. Nếu đối số trong lệnh HELLO không được chấp nhận, một reply failure 501 phải được trả về và receiver-SMTP đó phải ở trong cùng trạng thái.

Các lệnh NOOP, HELP, EXPN, và VRFY có thể được sử dụng vào bất kỳ thời điểm nào.

Các lệnh MAIL, SEND, SAML bắt đầu cho sự truyền mail. Khi được khởi động, sự truyền mail bao gồm một trong các lệnh khởi tạo, một hoặc nhiều lệnh RCPT và lệnh DATA. Sự truyền mail có thể bị huỷ bỏ bởi lệnh RSET. Có thể có nhiều hoặc không có sự truyền nào trong một phiên truyền.

Nếu đối số bắt đầu phiên truyền không được chấp nhận, thông báo 501 failure phải được trả về và reciever-SMTP phải nằm trong cùng trạng thái. Nếu các lệnh trong phiên truyền không có thứ tự, thì thông báo 503 failure sẽ được trả về và reciever-SMTP phải nằm trong cùng trạng thái.

Lệnh cuối cùng trong phiên truyền là lệnh QUIT. Lệnh này không thể được sử dụng tại bất kỳ thời gian nào trong phiên truyền.

2. *Cú pháp của các lệnh*

- Các lệnh bao gồm một mã lệnh theo sau là đối số của lệnh. Mã lệnh là 4 ký tự alphabetic. Không phân biệt chữ thường hoặc chữ hoa.
- Giữa mã lệnh và đối số là một hoặc nhiều khoảng trắng. Tuy nhiên trong reverse-path và forward-path, kiểu chữ rất quan trọng. Đặc biệt, trên một số host, tên user cũng phân biệt kiểu chữ hoa và thường.
- Đối số bao gồm một chuỗi ký tự có chiều dài biến đổi kết thúc bằng chuỗi ký tự " <CRLF> ".
- Dấu ngoặc vuông biểu diễn cho một vùng đối số tuỳ chọn.
- Sau đây là những lệnh SMTP:

HELO <SP> <domain> <CRLF>
MAIL <SP> FROM:<reverse-path> <CRLF>
RCPT <SP> TO:<forward-path> <CRLF>
DATA <CRLF>
RSET <CRLF>
SEND <SP> FROM:<reverse-path> <CRLF>
SOML <SP> FROM:<reverse-path> <CRLF>
SAML <SP> FROM:<reverse-path> <CRLF>
VRFY <SP> <string> <CRLF>
EXPN <SP> <string> <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>

QUIT <CRLF>
TURN <CRLF>

3. *Các reply của SMTP Server*

- Sự trả lời cho những lệnh của SMTP được đặt ra để đảm bảo cho sự đồng bộ cho các yêu cầu và những hoạt động trong quy trình truyền mail, và để bảo đảm rằng sender-SMTP luôn luôn biết trạng thái của reciever-SMTP. Mỗi lệnh SMTP phải tạo ra chính xác một reply.

- Một reply SMTP bao gồm một số ba chữ số (được truyền như ba ký tự chữ số) và theo sau là một số văn bản (text). Số đó được sử dụng một cách tự động để xác định trạng thái đưa vào kế tiếp. Text ở trên là dành cho người sử dụng. Ba chữ số đó được ấn định chứa đầy đủ thông tin được mã hoá mà sender-SMTP không cần kiểm tra text đó và có thể huỷ bỏ hay chuyển nó qua một user thích hợp. Đặc biệt text này có thể phụ thuộc vào receiver và vào ngữ cảnh, vì vậy có sự giống nhau trong sự phân biệt text cho từng mã reply.

❖ **Reply codes by function groups**

500 :Lỗi cú pháp, không nhậ dạng được lệnh.
501 :Lỗi cú pháp về thông số hoặc đối số.
503 : Chuỗi lệnh lỗi.
504 : Thông số lệnh không có.
211 : Trạng thái hệ thống, hay trả lời giúp đỡ về hệ thống
214 : Thông điệp giúp đỡ
220 :<domain> dịch vụ sẳn sàng
221 :<domain> dịch vụ đóng kênh truyền
421 :<domain> dịch vụ không dùng được, đóng kênh truyền
250 :Hành động mail yêu cầu OK, hoàn thành
251 : User không cục bộ, sẽ hướng đến “forward-path”
450 : Mail được yêu cầu không có, mailbox không tồn tại.
451 :Bỏ qua hành động được yêu cầu; lỗi trong quá trình xử lý
551 : User không cục bộ, thử lại <forward-path>
452 :Hành động được yêu cầu không thu được : hệ thống lưu trữ không đủ
552 :Bỏ qua hành động yêu cầu mail : vượt quá cấp phát lưu trữ
553 :Hành động được yêu cầu không chấp nhận : tên mailbox không cho phép [như sai cú pháp mailbox].
354 : Khởi động việc nhận mail; kết thúc với <CLRF>.<CLRF>
554 : Tryuền bị bị sai.

4. *Ví dụ về một giao dịch của SMTP*

1. Server : 220 sample2 Simple Mail Transfer Service Ready
khi được kết nối qua nghi thức TCP/IP, máy nhận trả lời với mã 220 đầu báo cho máy gởi biết dịch vụ SMTP đã sẵn sàng.
2. Client : HELLO tmt01vn
Bên nhận đã sẵn sàng, bên gởi gởi HELLO và xưng tên người gởi
3. Server : 250 hello.
Trả với mã 250 báo cho biết bên nhận đã sẵn sàng
4. Client : MAIL FROM:<tmt01vn@tmt01vn.com>

Bên gởi dùng lệnh MAIL để khởi động phiên giao dịch. Cú pháp như trên cho bên nhận biết địa chỉ bên gởi (mailbox của bên gởi) để bên nhận gởi thông báo lỗi nếu có về bên gởi

5. Server : 250 OK

Trả lời với mã 250 cho biết sẵn sàng

6. Client : RCPT TO:<phungkhn@tmt01vn.com>

7. Server: 250 OK

8. Client : RCPT TO: phungkhn1@yahoo.com

Muốn gởi cho bao nhiêu người dùng bấy nhiêu lệnh RCPT kèm theo địa chỉ nhận, bên nhận nếu đúng sẽ trả về mã 250 kèm theo OK

9. Server : 550 No such user here

Báo kèm theo mã 550 cho biết không có mailbox trên địa chỉ trên đối với nơi nhận

10. Client : DATA

Báo cho bên nhận biết dữ liệu bắt đầu từ sau từ DATA

11. Server : 354 Start mail input; end with <CRLF>.<CRLF>

Mã 354 báo cho biết đã sẵn sàng nhận mail, kết thúc mail với ký tự CRLF.CRLF

12. Client : Bắt đầu thân của mail

13. …v..v..

14. Client : (đến khi kết thúc nhấn CRLF.CRLF)

15. Server : 250 OK

16. Client : QUIT

Phát lệnh báo kết thúc phiên giao dịch

17. Server : 221 sample2 Service closing transmission channel

Mã 221 đóng kết nối đã thiết lập

Ví dụ trên sau phiên làm việc mail đ ược gởi tới địa chỉ mail phungkhn@yahoo.com

5. Nghi thức mở rộng ESMTP

- SMTP có một hạn chế gây khó khăn lớn trong việc truyền nhận mail là giới hạn tối đa kích thước nội dung một bức mail chỉ là 128KB. Ngày nay nội dung các bức mail không chỉ là dạng văn bản đơn thuần mà còn bao gồm hình ảnh, âm thanh và nhiều loại dữ liệu khác nữa, giới hạn 128KB trở nên quá nhỏ. Do vậy người ta đã cải tiến chuẩn SMTP thành một chuẩn mở rộng mới gọi là ESMTP.

- Chuẩn này cho phép tăng kích thước mail, nó đưa thêm từ khoá SIZE=nnnnnnnnn sau lệnh khởi động cuộc giao dịch, nhờ đó ta có thể tăng giới hạn kích thước của mail lên trên 1MB, đủ để chứa thêm vào các âm thanh, hình ảnh…

- Để biết xem Server MTA có theo chuẩn ESMTP hay không, thay vì dùng lệnh HELLO ở đầu một cuộc giao dịch, Client MTA dùng lệnh mới HELLO, nếu Server MTA có trang bị, nó sẽ trả về mã thành công là 250. Ngày nay chuẩn ESMTP đã thay thế chuẩn SMTP ở đa số các hệ thống.

Ví dụ : để khởi động cuộc giao dịch với kích thước mail lên tới 1MB, dòng lệnh sẽ là :

MAIL FROM :<thuan@sample1> SIZE=1000000

IV. GIAO THỨC POP3(RFC1081, RFC1082)

- Post Office Protocol Version 3 (Pop3) là một giao thức chuẩn trên internet cho phép một một workstation có thể truy xuất động đến một maildrop trên một server từ xa. Có nghĩa là Pop3 được dùng để cho phép workstation lấy mail mà server đang giữ nó.

- Port chuẩn dành cho dịch vụ Pop3 đươc qui ước là TCP port 110. Pop3 server sẽ khởi động và lắng nghe trên port này. Một client muốn sử dụng các dịch vụ của Pop3 thì nó phải thiết lập một kết nối tới Pop3 server. Khi kết nối được thiết lập thì Pop3 server sẽ gởi tới client một lời chào. Sau đó, Pop3 Client và Pop3 Server sau đó trao đổi các request và reply cho đến khi kết nối được đóng hay loại bỏ.

- Các lệnh trong Pop3 không phân biệt chữ thường và chữ hoa, bao gồm một tập từ khoá (chiều dài từ 3 đến 4 ký tự), có thể có hoặc không có đối số theo sau (chiều dài của đối số có thể lên đến 40 ký tự). Các từ khoá và đối số phân cách nhau bởi một ký tự trắng đơn, và không phải là các ký tự đặc biệt.

- Các reply trong Pop3 bao gồm phần chỉ định trạng thái và từ khoá có thể có các thông tin hỗ trợ theo sau. Chiều dài của reply có thể lên tới 512 ký tự, kết thúc bằng cặp CRLF. Có hai loại chỉ định trạng thái là: “+OK” và “-ERR”. Server phải gởi các chỉ định trạng thái ở dạng chữ hoa.

- Reply cho các lệnh có thể bao gồm nhiều dòng. Sau khi dòng đầu tiên và cặp ký tự CRLF được gởi đi, các dòng thêm vào được gởi đi, mỗi dòng kết thúc bằng một cặp CRLF. Dòng cuối là ký tự “.” và cặp ký tự CRLF. Nếu có dòng nào bắt đầu bằng ký tự “.” thì phải kiểm tra xem có phải là cặp ký tự kết thúc CRLF.

- Một Pop3 session sẽ phải trải qua các trạng thái: xác nhận (Authorization), giao dịch (transaction) và trạng thái cập nhật (Update).

- Trong trạng thái xác nhận, client phải thông báo cho server biết nó là ai. Khi server đã xác nhận được client, session sẽ đi vào trạng thái giao dịch. Trong trạng thái này, client hoạt động bằng cách gởi các request tới server. Khi client gởi lệnh “QUIT”, session sẽ đi vào trạng thái cập nhật (Update). Trong trạng thái này, Pop3 server giải phóng các tài nguyên và gởi lời tạm biệt. Sau đó kết nối TCP đóng lại.

- Các reply của Pop3 Server cho Pop3 client sẽ là “-ERR” nếu lệnh không nhận ra được bởi Pop3 server, hoặc không thực hiện được, hoặc sai cú pháp, hoặc sai trạng thái.

- Một Pop3 server có một khoảng thời gian time out. Khi xảy ra time out, session không đi vào trạng thái cập nhật (Update) mà server sẽ tự đóng kết nối TCP mà không xoá bất kỳ message nào hay gởi đáp ứng cho client.

1. Các trạng thái của pop3

Một khi kết nối TCP được mở ra bởi một Pop3 client. Pop3 server sẽ gởi lại cho Pop3 client một lời chào.

Ví dụ :

S: +OK POP3 server ready

a.Trạng thái xác nhận (authorization):

- Sau khi Pop3 server gởi lời chào, session sẽ đi vào trạng thái xác nhận (authorization). Lúc này, Pop3 client phải định danh và xác nhận nó với

Pop3 server. Để thực hiện việc này, client phải sử dụng kết hợp các lệnh USER và PASS.

- Đầu tiên, client sẽ gởi lệnh "USER username", nếu Pop3 server trả lời với chỉ thị trạng thái "-ERR" thì client có thể đưa ra một lệnh xác nhận mới hay có thể đưa ra lệnh "QUIT".

- Nếu Pop3 server trả lời với chỉ thị trạng thái "+OK", thì client có thể gởi tiếp lệnh "PASS password" để hoàn tất sự xác nhận hoặc gởi lệnh "QUIT" để kết thúc session.

- Khi client phát ra một lệnh "PASS", POP3 server dùng cặp đối số từ lệnh USER và PASS để xác định nếu đúng client sẽ cho truy xuất đến maildrop thích hợp.

Sau khi trải qua quá trình xác nhận, Pop3 server sẽ cho phép client truy xuất tới những mailbox thích hợp. Lúc này, Pop3 server sẽ tạo ra một khoá truy xuất loại trừ trên maildrop để đảm bảo cho message không bị sửa đổi hay bị xoá trước khi session đi vào trạng thái cập nhật (Update). Nếu thành công, Pop3 server sẽ trả lời với chỉ thị trạng thái "+OK" và session sẽ đi vào trạng thái giao dịch (transaction) mà không có message bị đánh dấu xoá. Nếu maildrop không mở được vì một lý do nào đó (ví dụ: sai khoá, client bị từ chối truy xuất tới maildrop này), Pop3 server sẽ trả lời với chỉ thị trạng thái "-ERR" và server sẽ đóng kết nối. Nếu kết nối không bị đóng thì client có thể gởi lệnh xác nhận mới và bắt đầu trở lại hoặc có thể phát ra lệnh "QUIT".

- Sau khi Pop3 server mở được maildrop, nó gán số thứ tự cho mỗi message và biểu thị kích thước message theo byte.

- Chú ý rằng, đây là thông tin phản hồi từ phía POP3 Server. Dấu "+" có nghĩa là thành công, ngược lại, dấu "-" là không thành công bị lỗi. Phiên làm việc POP3 hiện tại đang ở trạng thái AUTHORIZATION. Phía Client bây giờ cần phải đưa vào các lệnh để xác định người nhận thư cho POP3 Server.

b.Trạng thái giao dịch (transaction):

Sau khi Pop3 server đã xác nhận thành công client, và mở cho nó một maildrop thích hợp. Session sẽ bước vào trạng thái giao dịch (transaction). Lúc này, Pop3 client có thể gởi các request cho Pop3 server (các request có thể được gởi nhiều lần, tức là có thể lặp lại), và cứ sau mỗi request thì Pop3 server sẽ phản hồi lại cho Pop3 client một reply. Cuối cùng, nếu client phát ra lệnh "QUIT" thì session sẽ đi vào trạng thái cập nhật (Update).

c. Trạng thái cập nhật (Update):

Khi client phát ra lệnh "QUIT" từ trạng thái giao dịch (transaction), session sẽ đi vào trạng thái cập nhật (Update). Nếu client phát ra lệnh "QUIT" từ trạng thái xác nhận (authorization), session sẽ kết thúc nhưng không đi vào trạng thái cập nhật.

Nếu session kết thúc vì các lý do khác sau đó một lệnh "QUIT" được phát ra từ client, session sẽ không đi vào trạng thái cập nhật (Update) và phải không xoá một message nào từ maildrop.

2. Các lệnh của POP3:

a. Các lệnh có tác dụng trong quá trình xác nhận (authorization):

♦ **USER** username:

+ Đối số username là một chuỗi định danh một mailbox, chỉ có ý nghĩa đối với server.

+ Trả lời: +OK tên mailbox có hiệu lực.

-ERR không chấp nhận tên mailbox.

♦ **PASS** string:

+ Đối số là một password cho mailbox hay server.

+ Trả lời: +OK khoá maildrop và sẵn sàng.

-ERR password không hiệu lực.

-ERR không được phép khoá maildrop.

b. Các lệnh có tác dụng trong quá trình giao dịch (transaction):

♦ **STAT**:

+ Không có đối số.

+ Trả lời: +OK nn mm. "+OK" theo sau là khoảng trắng đơn, tiếp theo là nn: số message, khoảng trắng đơn, mm: kích thước của maildrop tính theo byte.

+ Các message được đánh dấu xoá không được đếm trong tổng số.

♦ **LIST** [msg]:

+ Đối số: số thứ tự của message, có thể không tham khảo tới các message đã được đánh dấu xoá.

+ Trả lời: +OK scan listing follow.

-ERR nosuch message.

Một scan listing bao gồm số thứ tự message (message number) của message đó, theo sau là khoảng trắng đơn, và kích thước chính xác của message đó tính theo byte.

♦ **RETR** msg:

+ Đối số: số thứ tự của message, có thể không tham khảo tới các message đã được đánh dấu xoá.

+ Trả lời: +OK message follows

-ERR no such message

Trả lời của lệnh RETR là multi-line.

♦ **DELE** msg:

+ Đối số: số thứ tự của message, có thể không tham khảo tới các message đã được đánh dấu xoá.

+ Trả lời: +OK message deleted

-ERR no such message

Pop3 server sẽ đánh dấu xoá các message này. Tuy nhiên, quá trình xoá thật sự sẽ diễn ra ở trạng thái cập nhật (Update).

♦ **NOOP**:

+ Không có đối số.

+ Trả lời: +OK

Pop3 server không làm gì hết, chỉ hồi âm lại cho client với trả lời: "+OK".

♦ **RSET**:

+ Không có đối số.

+ Trả lời: +OK.

Phục hồi lại các message đã bị đánh dấu xoá bởi Pop3 server.

♦ **QUIT**:
+ Không có đối số.
+ Trả lời: +OK.

3. Ví dụ về một session của Pop3:

Giai đoạn 1 : Nhận dạng user

CLIENT : USER Tuyentm // cho biết tên user là Tuyentm
SERVER : +OK // báo thành công
CLIENT : PASS kimphung // cho biết password là tin
SERVER : +OK complet: maildrop has 2 messages (520 octets…)

Giai đoạn 2 : Trao đổi

CLIENT : STAT // số mail có trong mailbox
SERVER : +OK 2 520 // có 2 mail với tổng kích thước là 520
CLIENT : LIST // Liệt kê các ID và kích thước các mail
SERVER : +OK 2 message (520 octets)
SERVER : 1 110 // mail thứ 1 kích thước 110
SERVER : 2 410 // mail thứ 2 kích thước 410
CLIENT : LIST 1 // Cho thông tin về mail có ID là 1
SERVER : +OK 1 110
CLIENT : LIST 4
SERVER : -ERR nosuch message, only 2 message in maildrop
….v…v…

Giai đoạn 3 :

CLIENT : QUIT ; đóng kết nối TCP hiện hành
SERVER : +OK dhbk POP3 server signing off…

Chú ý rằng các message bị đánh dấu để xoá bằng lệnh DELE thực sự chưa bị xoá ngay để nếu sau đó ta có thể dùng lệnh phục hồi không xoá bằng lệnh RSET, chúng chỉ thực sự bị xoá bỏ khỏi maildrop khi bước vào giai đoạn Update (khi gởi lệnh QUIT).

V. GIAO THỨC IMAP4(RFC2060, RFC2193…)

- Internet Message Access Protocol (IMAP) cung cấp lệnh để phần mềm thư điện tử trên máy khách và máy chủ dùng trong trao đổi thông tin phiên bản 4(IMAP4rev1). Đó là phương pháp để người dùng cuối truy cập thông điệp thư điện tử hay bản tin điện tử từ máy chủ về thư trong môi trường cộng tác. Nó cho phép chương trình thư điện tử dùng cho máy khách - như Netscape Mail, Eudora của Qualcomm, Lotus Notes hay Microsoft Outlook - lấy thông điệp từ xa trên máy chủ một cách dễ dàng như trên đĩa cứng cục bộ.
- IMAP khác với giao thức truy cập thư điện tử Post Office Protocol (POP). POP lưu trữ toàn bộ thông điệp trên máy chủ. Người dùng kết nối bằng đường điện thoại vào máy chủ và POP sẽ đưa các thông điệp vào in-box của người dùng, sau đó xoá thư trên máy chủ. Hai giao thức này đã được dùng từ hơn 10 năm nay. Theo một nhà phân tích thì khác biệt chính giữa POP (phiên bản hiện hành 3.0) và IMAP (phiên bản hiện hành 4.0) là POP3 cho người dùng ít quyền điều khiển hơn trên thông điệp.

- IMAP4rev1 được kế thừa từ [IMAP2] tuy nhiên trong giao thức IMAP4rev1 không tồn tại các giao thức hay cấu trúc của [IMAP2] nhưng những khuôn dạng dữ liệu vẫn được kế thừa và sử dụng. IMAP4rev1 bao gồm những thao tác tạo ra, xoá, và đổi tên các hòm thư, kiểm tra mail mới, thường xuyên cập nhật lại cờ những mail cũ nhưng thao tác này được trình bày trong RFC822(RFC dùng chuẩn hoá message) và những thao tác này là duy nhất.

- IMAP là cơ chế cho phép lấy thông tin về thư điện tử của bạn, hay chính các thông điệp từ mail server của môi trường cộng tác. Giao thức thư điện tử này cho phép người dùng kết nối bằng đường điện thoại vào máy chủ Internet từ xa, xem xét phần tiêu đề và người gửi của thư điện tử trước khi tải những thư này về máy chủ của mình. Với IMAP người dùng có thể truy cập các thông điệp như chúng được lưu trữ cục bộ trong khi thực tế lại là thao tác trên máy chủ cách xa hàng ki lô mét. Với khả năng truy cập từ xa này, IMAP dễ được người dùng cộng tác chấp nhận vì họ coi trọng khả năng làm việc lưu động.

- Một kết nối của IMAP4rev1 được thành lập theo một kết nối Client/Server và sự tương tác trao đổi thông tin hay lấy mail về từ Server của người sử dụng thông qua các lệnh truy suất mà IMAP4rev1 đã định dạng sẵn trong giao thức IMAP. người sử dụng bắt đầu một mã lệnh trong giao thức IMAP theo một quy luật là : đầu mỗi câu lệnh thêm vào các ký tự tượng trưng (nó tượng trưng cho lý lịch hay thứ tự của lệnh…) như khi gởi lệnh Login trong giao thức IMAP phải là *0001 Login Tuyen minhtuyen*.

1. *Các lệnh của IMAP4:*

- Những tập lệnh của IMAP4rev1 được định nghĩa trong rfc2060 cũng nhưng quá trình bắt đầu và kết thúc của một phiên làm việc. Vì trong chương trình em chỉ sử dụng một số lệnh cơ bản trong bộ giao thức này, dưới đây là ý nghĩ cũng như cách sử dụng chúng.

♦ **CAPABILITY**

- Arguments: none
- Kết quả trả về : OK - capability completed
  BAD - command unknown or arguments invalid
- Đây là lệnh thực hiện trước tiên của bất kỳ một trình mail Client nào muốn lấy mail từ trình chủ bằng giao thức IMAP, mục đích là kiểm tra version giao thức có đáp ứng được yêu cầu không. Version hiện nay đang dùng là IMAP4(IMAP4rev1).

Ví dụ C: abcd CAPABILITY
S: * CAPABILITY IMAP4rev1
S: abcd OK CAPABILITY completed

♦ **LOGIN**

- Arguments: [user name] [password]
- Kết quả trả về là: OK - login completed, now in authenticated state
  NO - login failure: user name or password rejected
  BAD - command unknown or arguments invalid
- Lệnh này để xác nhận người sử dụng có hợp pháp không? Nếu thành công thì người dùng sẽ thực hiện các thao tác lệnh tiếp theo.

Ví dụ C: a001 LOGIN tuyentm01 kimphung
S: a001 OK LOGIN completed

♦ **CHECK**

- Arguments: none
- Kết quả trả về: OK - check completed
  BAD - command unknown or arguments invalid
- Lệnh này dùng để kiểm tra tại thời điểm này lệnh SELECT đã thực hiện hay chưa, nếu thực hiện rồi trả về OK.

♦ **SELECT**

- Arguments: mailbox name (tên hòm thư)
- Kết quả trả về : OK - select completed, now in selected state
  NO - select failure, now in authenticated state: no such mailbox, can't access mailbox
  BAD - command unknown or arguments invalid
- Lệnh Select dùng để nhận biết được hòm thư có bao nhiêu thư bao gồm thư mới, thư đọc rồi và thư đã xoá. Lệnh này cho phép ta thay đổi thuộc tính của hòm thư cũng như nhưng lá thư mà chúng lưu trữ bởi các lệnh khác trong IMAP.

Ví dụ C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited
S: A142 OK [READ-WRITE] SELECT completed.

- Trong ví dụ trên chúng ta quan tâm các thông số sau:

✓ EXISTS : tổng số lá thư mà hòm thư này lưu trữ ví dụ trên là 172 lá thư.

✓ RECENT : là số lá thư mới trong thời gian gần đây mà người sử dụng chưa đọc ví dụ trên là 1.

✓ UNSEEN : là tổng số lá thư củ mà người dùng chỉ nhìn thấy nhưng nội dung chưa xem qua.

✓ UIDVALIDITY : dùng để chỉ định trạng thái của hòm thư đây là một thông số không quan trong.Mỗi mail Server sẽ có cách đặc tả thông số này khác nhau tuỳ từng mục đích sử dụng nó của các nhà quản trị mail thông số này liên quan đến lệnh UID.

♦**CLOSE**

- Arguments: none
- Kết quả trả về : OK - close completed, now in authenticated state
  NO - close failure: no mailbox selected
  BAD - command unknown or arguments invalid
- Lệnh này dùng để đóng lệnh SELECT lại hay có thể hiểu loại bỏ lệnh này và không lưu lại các thuộc tính đã thay đổi với hòm thư này.

♦ **FETCH**

- Arguments: message set message data item names
- Kết quả: OK - fetch completed
  NO - fetch error: can't fetch that data
  BAD - command unknown or arguments invalid

- Lệnh dùng để hiển thị nội dung của một lá thư. Thông số theo sau gồm có hai thông số: đầu tiên là số thứ tự của lá thư và thông số thư hai là message data item names nhưng thông số này phải tuân theo RFC822 được trình bày ở trên.

Ví dụ: C: A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])
S: * 2 FETCH ....
S: * 3 FETCH ....
S: * 4 FETCH ....
S: A654 OK FETCH completed

♦ **UID**

- Arguments: là các lệnh trong IMAP
- Kết quả trả về: OK - UID command completed
  NO - UID command error
  BAD - command unknown or arguments invalid

♦ **EXAMINE**

- Arguments: mailbox name
- Kết quản trả về: OK - examine completed, now in selected state
  NO - examine failure, now in authenticated state: no such mailbox, can't access mailbox
  BAD - command unknown or arguments invalid

- Lệnh này tương tự như lệnh SELECT cùng một kế quả trả về nhưng khi dùng lệnh này chúng ta chỉ xem thông tin không thay đổi được trạng thái của hòm thư cũng như các thuộc tính của nó.

Ví dụ: C: A932 EXAMINE Inbox
S: * 17 EXISTS
S: * 2 RECENT
S: * OK [UNSEEN 8] Message 8 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS ()] No permanent flags permitted
S: A932 OK [READ-ONLY] EXAMINE completed

♦ **CREATE**

- Arguments: tên hòm thư cần tạo.
- Kết quả trả về:OK - create completed
  NO - create failure: can't create mailbox with that name
  BAD - command unknown or arguments invalid

- Lênh tạo ra một hòm thư mới với tên đã chọn và trả lại là OK nếu quá trình tạo ra hòm thư trên Server không gặp lỗi.

Ví dụ: C: A003 CREATE Tuyen
S: A003 OK CREATE completed
C: A004 CREATE Inbox
S: A004 No mailbox name Exist

♦ **DELETE**
- Arguments: tên hòm thư cần xoá.
- Kết quả trả về:OK - delete completed
NO - delete failure: can't delete mailbox with that name
BAD - command unknown or arguments invalid
- Lệnh xoá hòm thư, nếu xoá thành công thì kết quả nhận được là OK.

Ví dụ: C: A682 LIST "" *
S: * LIST () "/" Inbox
S: * LIST () "/" Tuyen
S: A682 OK LIST completed
C: A683 DELETE Tuyen
S: A683 OK DELETE completed
C: A684 DELETE Tuyen
S: A684 NO Name "Tuyen" has inferior hierarchical names
C: A686 LIST "" *
S: * LIST () "/" Inbox
S: A686 OK LIST completed

♦ **RENAME**
- Arguments: (tên hòm thư tồn tại) (tên hòm thư mới).
- Kết quả: OK - rename completed
NO - rename failure: can't rename mailbox with that name, can't rename to mailbox with that name
BAD - command unknown or arguments invalid
- Lệnh chuyển đổi tên hòm thư, kết quả là OK nếu thành công.

♦ **COPY**
- Arguments: tên lá thư đến tên hòm thư
- Kết quả trả về:OK - copy completed
NO - copy error: can't copy those messages or to that name
BAD - command unknown or arguments invalid
- Đây là lệnh copy một lá thư từ hòm thư này sang hòm thư khác.

Ví dụ: C: A003 COPY 2:4 MEETING
S: A003 OK COPY completed

♦ **SUBSCRIBE**
- Arguments: tên hòm thư
- Kết quả trả về: OK - subscribe completed
NO - subscribe failure: can't subscribe to that name

BAD - command unknown or arguments invalid

- Lệnh dùng để thiết lập thuộc tính active của hòm thư,tuy nhiên nó không thể thay đổi được đặc tính hòm thư hay nói cách khác nó dùng để kiểm tra xem hòm thư này có tồn tại hay không

♦ **UNSUBSCRIBE**

- Arguments: tên hòm thư
- Kết quả trả về:OK - unsubscribe completed
  NO - unsubscribe failure: can't unsubscribe that name
  BAD - command unknown or arguments invalid

- Lệnh này ngược lại với SUBSCRIBE nghĩa là nó loại bỏ thuộc tính active của hòm thư.

♦ **LIST**

- Arguments: tên hay những ký tự đặc trưng.
- Kết quả trả về:OK - list completed
  NO - list failure: can't list that reference or name
  BAD - command unknown or arguments invalid

- Nếu tên hay những ký tự theo sau hợp lệ thì lệnh này trả về tập tên các hòm thư, thường tên hay ký tự theo sau là “% ,*, “”, /, String*,”. Chúng ta tạm hiểu nó như một lệnh Dir trong MS-DOS.

Ví dụ: C: A101 LIST "" ""
S: * LIST (\Noselect) "/" ""
S: A101 OK LIST Completed

C: A102 LIST Tuy* ""
S: * LIST (\Noselect) "/" ""
S: A102 OK LIST Completed

C: A102 LIST * *
S: * LIST () "/*" "Inbox"
S: * LIST () "/*" "Tuyen"
S: * LIST () "/*" "Phung"
S: A102 OK LIST Completed

C: A102 LIST Tuy* *
S: * LIST () "tuy*" "tuyen"
S: A102 OK LIST Complete

♦ **LUSB**

- Arguments: tên hay những ký tự đặc trưng.
- Kết quả trả về: OK - list completed
  NO - list failure: can't list that reference or name
  BAD - command unknown or arguments invalid

Lệnh này tương tự như list nhưng chỉ khác một điều là nhưng hòm thư nhận được phải ở trạng thái active.

♦ **STATUS**

- Arguments: tên hòm thư (trạng thái)

- Kết quả trả về:OK - status completed
NO - status failure: no status for that name
BAD - command unknown or arguments invalid

- Lệnh này trả về trạng thái hiện tại của hòm thư, nó không làm ảnh hưởng đến sự chuyển đổi của hòm thư cũng như các trạng thái của các lá thư. Trạng thái theo sao hiện nay trong IMAP4rev1 như sau.chức năng của lệnh này dùng để check mail.

✓ MESSAGES : số thư mới trong hòm thư
✓ RECENT : số lá thơ củ.
✓ UIDNEXT : giá trị UID tiếp theo sẽ được gán cho một lá thư mới trong hòm thư
✓ UIDVALIDITY : giá trị UID của hòm thư.
✓ UNSEEN : nhưng lá thư của mà người dùng chưa xem nội dung.

Ví dụ C: A042 STATUS tuyen (MESSAGES RECENT)
S: * STATUS tuyen (MESSAGES 23 RECENT 40)
S: A042 OK STATUS completed

♦ **NOOP**

- Arguments: none
- Kết quả trả về: OK - noop completed
BAD - command unknown or arguments invalid

- Lệnh này thực chất không làm gì cả mà mục đích để kiểm tra xem giữa mail Client và mail Server còn liên lạc với nhau không.

Ví dụ C: a002 NOOP
S: a002 OK NOOP completed
. . .
C: a047 NOOP
S: * 22 EXPUNGE
S: * 23 EXISTS
S: * 3 RECENT
S: * 14 FETCH (FLAGS (\Seen \Deleted))
S: a047 OK NOOP completed

♦ **STORE**

Arguments: message set message data item name value for message data item
Responses: untagged responses: FETCH
Result: OK - store completed
NO - store error: can't store that data
BAD - command unknown or arguments invalid

- Thiết lập lại trạng thái của thư, mỗi lá thư có nhưng trạng thái như thư mới nhận là Recent tiếp theo là trạng thái chưa đọc Unsen, đọc rồi Seen và trạng thái xoá Deleted và một số cờ đặt trưng khác.
- Nếu thêm trạng thái thì dùng lệnh trong Arguments tương ứng là +FLAGS <flag list> hay +FLAGS.SILENT <flag list>

- Nếu loại bỏ trạng thái thì dùng lệnh trong Arguments tương ứng là -FLAGS <flag list> hay -FLAGS.SILENT <flag list>

- FLAGS.SILENT là thiết lập lại trạng thái server hồi đáp lại là hiên tại lá thư đó đang ở những trạng thái nào lệnh này ngược lại với lệnh –FLAGS

ví dụ

C: A003 STORE 2:4 +FLAGS (\Deleted)
S: * 2 FETCH FLAGS (\Deleted \Seen)
S: * 3 FETCH FLAGS (\Deleted)
S: * 4 FETCH FLAGS (\Deleted \Flagged \Seen)
S: A003 OK STORE completed
C: A003 STORE 2:4 +FLAGS.SILENT (\Deleted)
S: A003 OK STORE completed

♦ **EXPUNGE Command**

Arguments: none
Responses: untagged responses: EXPUNGE
Result : OK - expunge completed
NO - expunge failure: can't expunge (e.g. permission denied)
BAD - command unknown or arguments invalid

Lệnh dùng để kiểm tra những lá thư có trạng thái deleted và loại nó ra khỏi hòm thư, đưa vào thùng rác(hòm thư trash). Nếu những như hòm thư Trash được chọn thì những lá thư này sẽ được xoá ra khỏi mail của bạn.

♦ **LOGOUT**

- Arguments: none
- Kết quả tả về: OK - logout completed
BAD - command unknown or arguments invalid
- Lệnh dùng để đóng kết nối lại sao một phiên làm việc.

Ngoài các lệnh trên trong IMAP4 còn một số lệnh khá hay khác như SEARCH, AUTHENTICATE,... vì thời gian và năng lực có hạn nên đồ án của em có lẽ chỉ dừng lại ở các lệnh trên.

CHƯƠNG 4

GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH JAVA.

I.Tổng quan về ngôn ngữ lập trình Java.

1. ***Sự xuất hiện ngôn ngữ Java:***

- Một thực tế công nhận rằng, Java ngôn ngữ lập trình của Sun Microsystems, có một sức mạnh đầy ấn tượng và là chủ đề đang được tranh luận nhiều nhất hiện nay, Logo của Java- một tách cafe bốc khói, cùng các Applet Java đang tràn ngập khắp World Wide Web, và được các ngành công nghiệp chấp nhận với một tốc độ chưa từng có. Vậy thực ra Java là gì? Uy lực của chúng ra sao? Đó là tất cả những gì tôi muốn nói cùng các bạn, trong quá trình làm thực tập tôi đã lượm lặt được.

- Java là một ngôn ngữ lập trình do công ty Sun Microsystems phát triển vào đầu thập kỷ 1990. Xuất phát điểm của ngôn ngữ này là một dự án nghiên cứu của công ty dựa trên nền tảng C++. Vì Java xuất phát từ một dự án nghiên cứu chứ không phải là một sản phẩm nhằm mục đích sinh lời, công ty Sun đã đồng ý đưa Java Development KIT (Bộ công cụ phát triển Java, bao gồm chương trình dịch và hệ thống đáp ứng chạy chương trình) lên internet miễn phí vào khoảng giữa năm 1995.

- Sự xuất hiện của Java và các trình duyệt web hiểu Java (chạy được Java applets) đã giải quyết được nhược điểm của WSA (Web Serever Application), hỗ trợ cho lập trình với Socket, và mở ra một hướng mới. Ưu điểm của Java Applet là chúng được truyền trong mạng, và chạy trong trình duyệt web. Vì vậy, dữ liệu có thể được truyền thông qua chúng, chứ không phải dưới dạng văn bản. Chỉ bằng cách tạo thêm Java client cho các WSA đang hoạt động, trao đổi thông tin giữa trình duyệt web và WSA, tốc độ thực hiện của các ứng dụng này đã nhanh lên rất nhiều, và cho phép sử dụng chúng trên một thời gian nữa.

- Hệ thống Java bao gồm một số cấu phần như sau: Ngôn ngữ lập trình Java, Java Virtual Machine (Máy ảo Java, bộ phận thông dịch), các thư viện phần mềm đi kèm hệ thống, chương trình duyệt web hot Java hoặc chương trình duyệt web khác thích ứng với Java (với Micrsoft Explorer, Netscape…).

- Thực ra Java không đưa ra nhiều ý tưởng hoàn toàn mới mà sự đổi mới ở đây là việc gom lại các ý tưởng hay nhất của các ngôn ngữ đi trước. Java là một ngôn ngữ lập trình hướng đối tượng trong cùng một họ như C++, Pascal,

Algol 60. Các chương trình Java có các mô tả dữ liệu và các câu lệnh nhóm lại trong các hàm (do Java là một ngôn ngữ lập trình hướng đối tượng, các hạn của nó thường được gọi là các "Phương pháp" (method). Các phương pháp có thể gọi tới các phương pháp khác. Sự thực hiện các chương trình sẽ được bắt đầu với một phương pháp có tên đặc biệt là main). Mặc dù Java đã bổ sung một số điểm mới, nét phân biệt nhất của Java là những đặc điểm mà nó bỏ qua. Chẳng hạn, so sánh với C, Java không có.

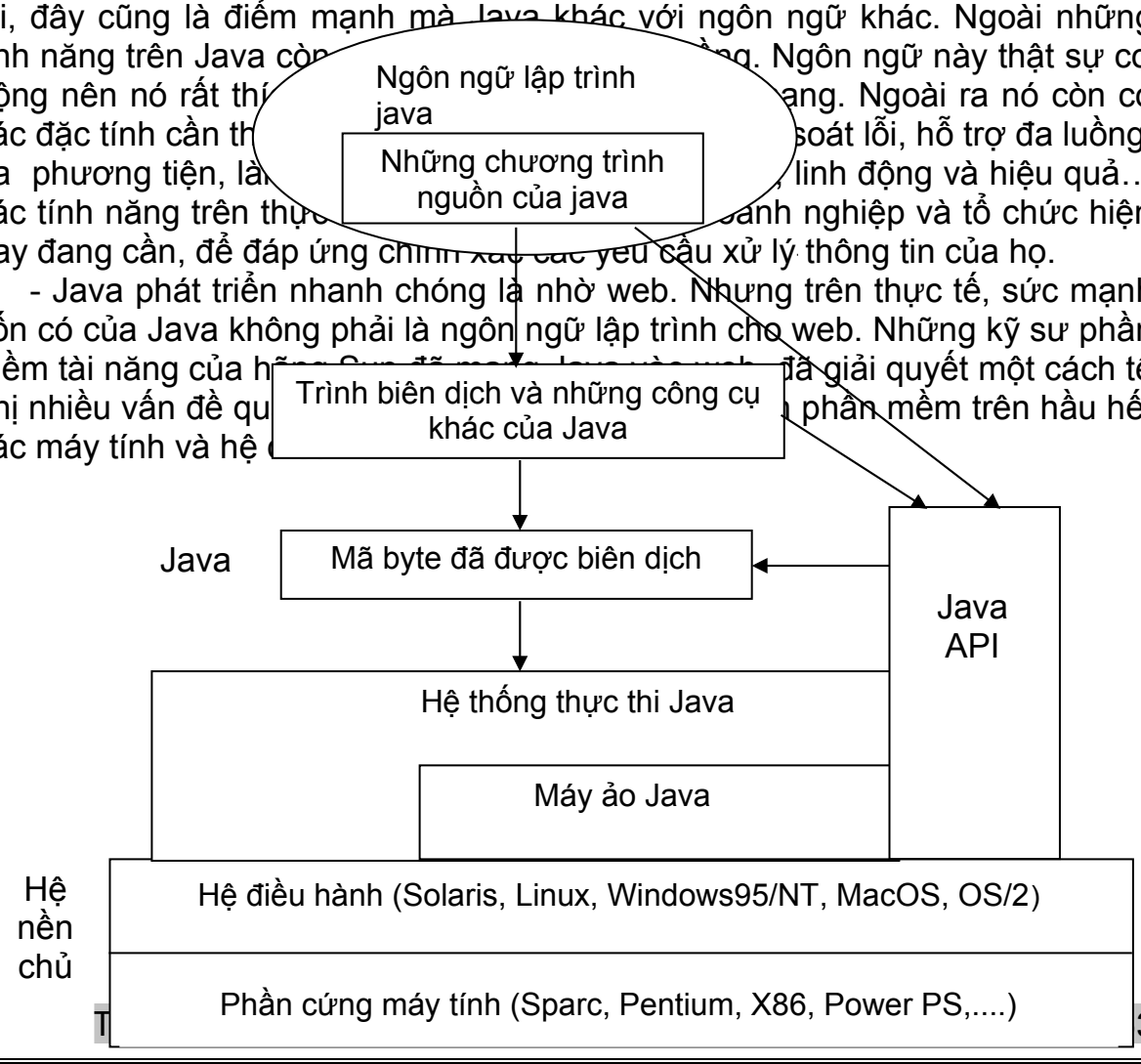
- Số học địa chỉ bộ nhớ (con trỏ)
- Bộ tiền xử lý
- Câu lệnh goto
- Chuyển đổi kiểu tự động
- Các biến và hàm tổng thể
- Các định danh xác định kiểu.

Còn so với C++ thì Java không có

- Các mẫu (template – tuy nhiên người ta đang tính đến việc bổ sung đặc điểm này cho Java)
- Quá tải toán tử (operator overloading)
- Đa thừa kế
- Đa ABI (Appliacation Binary Interface, giao diện nhị phân ứng dụng)

- Giống như hầu hết những ngôn ngữ lập trình hướng đối tượng khác, Java bao gồm: một ngôn ngữ, một môi trường, một giao diện ứng dụng Java và nhiều lớp thư viện. Những lớp riêng này có đặc tính riêng là tái sử dụng lại, đây cũng là điểm mạnh mà Java khác với ngôn ngữ khác. Ngoài những tính năng trên Java còn ng. Ngôn ngữ này thật sự cơ động nên nó rất thí ang. Ngoài ra nó còn có các đặc tính cần th soát lỗi, hỗ trợ đa luồng, đa phương tiện, là linh động và hiệu quả… các tính năng trên thực oanh nghiệp và tổ chức hiện nay đang cần, để đáp ứng chính xác các yêu cầu xử lý thông tin của họ.

- Java phát triển nhanh chóng là nhờ web. Nhưng trên thực tế, sức mạnh vốn có của Java không phải là ngôn ngữ lập trình cho web. Những kỹ sư phần mềm tài năng của hãng Sun đã mang Java vào web đã giải quyết một cách tế nhị nhiều vấn đề qu phần mềm trên hầu hết các máy tính và hệ



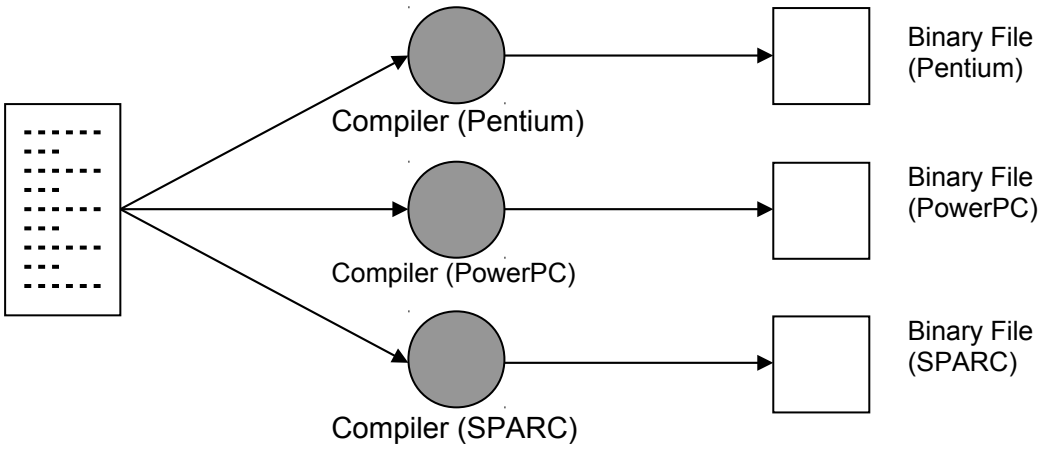
Hình 4.1 Bức màn bí mật của Java

- Bạn nên hiểu về Java là ngoài việc tạo các Applet, bạn có thể sử dụng Java vào nhiều ứng dụng khác. Java được thiết kế như một ngôn ngữ lập trình hoàn chỉnh, với Java bạn có thể tiến hành lập trình như với mọi ngôn ngữ lập trình khác như : Pascal, C, VisualBasic … . Bản thân Hot Java cũng được viết bằng chính Java.

- Java là một môi trường độc lập, đó là một trong những lợi thế quan trọng cho phép Java hơn hẳn những ngôn ngữ khác, đặc biệt cho những hệ thống cần để làm việc trên nhiều môi trường (máy tính, hệ điều hành) khác nhau. Java là một môi trường độc lập ở cả trên nền hệ thống lẫn dưới mức thấp như xử lý nhị phân.

- Môi trường độc lập là một chương trình có khả năng duy chuyển dễ dàng từ hệ thống máy tính này sang hệ thống máy tính khác không phụ thuộc vào cấu trúc của máy hay hệ điều hành hoạt động trên máy.

- Ở mức nền, các kiểu dữ liệu cơ bản của Java có vị trí vững chắc trong tất cả các môi trường. Những lớp thư viện của Java giúp cho bạn có thể viết ra các chương trình mà nó có thể đem từ máy tính này sang máy tính khác mà không cần phải viết lại để chạy riêng cho máy mới.



Hình 4.2 Biên dịch Java

- Môi trường độc lập không bao giờ chỉ dừng lại ở mức nền. Tập tin chứa mã Java đã dịch cũng giống như môi trường độc lập, nó cũng có thể chạy trên nhiều loại máy tính khác nhau mà không gặp phải vấn đề đòi hỏi biên dịch lại chương trình nguồn. Vậy nó làm việc như thế nào ? Những mã Java thực sự được gọi là Bytecode. Với bytecode là một tập hợp các chỉ dẫn trông khá giống mã máy, nhưng nó không cụ thể cho một bộ xử lý nào. Thông thường, khi bạn biên dịch một chương trình viết trong C hay hầu hết những ngôn ngữ khác. Chương trình biên dịch sẽ dịch chương trình của bạn ra mã máy hoặc các chỉ thị cho bộ xử lý. Những chỉ thị này hoàn toàn cụ thể đối với từng bộ xử lý để máy tính của bạn chạy được nó - vì vậy, ví dụ khi bạn biên dịch chương trình của bạn trên một máy Pentium, kết quả chương trình sẽ chỉ chạy được trên những hệ thống Pentium khác. Nếu bạn muốn dùng cùng chương trình này trên một hệ thống khác (ví dụ như Applet), bạn phải sửa lại chương trình nguồn của bạn, lấy trình biên dịch cho hệ thống mới này và biên dịch lại cho chương trình nguồn của bạn. Chương trình phải biên dịch lại khi muốn thực thi trên hệ thống khác.

- Mọi thứ sẽ trở nên khác khi bạn viết chương trình trong Java. Môi trường phát triển Java gồm có hai phần : một Java Compiler (Chương trình biên dịch Java) và một Java interpreter (chương trình thông dịch Java). Chương trình biên dịch Java sẽ làm việc với chương trình Java của bạn và thay vì phát sinh ra mã máy từ tập tin chứa chương trình nguồn của bạn, nó lại phát sinh ra mã bytecode.

- Để chạy một chương trình Java, bạn gọi một một chương trình có tên là bytecode interpreter, nó sẽ thực thi chương trình Java của bạn. Bạn có thể chạy bằng chương trình thông dịch (interpreter) hoặc là (đối với applet) bằng một bytecode interpreter cài sẵn trong HotJava hay trong một chương trình duyệt xét có tính năng Java khác để chạy những applet cho bạn. Chương trình Java viết dưới dạng bytecode có nghĩa là thay vì được dịch cụ thể với từng hệ thống một, chương trình của bạn sẽ có được khả năng chạy trong nhiều loại máy tính và hệ điều hành khác nhau khi bộ thông dịch Java còn có hiệu lực ở đó. Khả năng này của tập tin mã Java sẽ giúp cho nó thực thi được trên các loại môi trường khác nhau và là tính cốt yếu với những gì applet có thể làm việc được, bởi vì World Wide Web tự nó cũng là môi trường độc lập. Trong lúc tập tin HTML có thể được đọc trong nhiều môi trường, cũng như applet có thể thực thi trong những môi trường mà những chương trình duyệt xét có tính năng Java chạy được trên đó.

- Yếu điểm của việc dùng bytecode là trong tốc độ thực thi. Bởi vì với những chương trình chạy trong một hệ thống cụ thể, việc chạy trực tiếp trên những phần cứng cho những cái mà nó sẽ thực thi bởi chương trình thông dịch. Vì đối với nhiều chương trình Java, tốc độ không còn là vấn đề quan trọng. Nếu bạn viết những chương trình mà cần nhiều tốc độ thực thi hơn, bộ thông dịch Java có thể cung cấp bạn có một vài phương án thích hợp với bạn, bao gồm việc có thể liên kết những chương trình nguồn vào trong chương trình Java của bạn hoặc là dùng những công cụ chuyển đổi bytecode trở lại thành chương trình nguồn. Chú ý rằng nếu bạn dùng những phương án này, bạn sẽ mất tính chất năng động mà Java bytecode đã cung cấp cho bạn.

- Đối với một vài người, kỹ thuật lập trình hướng đối tượng (object - oriented - programming) chỉ đơn thuần là một cách tổ chức chương trình, và nó có thể sử dụng tốt trong nhiều ngôn ngữ. Khi làm việc với ngôn ngữ lập trình hướng đối tượng thực và môi trường lập trình này, bạn có khả năng đạt được hết tất cả những lời thế của phương pháp lập trình hướng đối tượng và khả năng để tạo ra sự mềm dẻo, tính linh hoạt, lập trình theo từng khối và sử dụng lại chương trình cũ. Giống như hầu hết những ngôn ngữ lập trình hướng đối tượng khác, Java bao gồm một tập các lớp thư viện mà nó cung cấp những kiểu dữ liệu cơ bản, dữ liệu nhập của hệ thống, khả năng xuất dữ liệu và những hàm tiện ích. Những lớp cơ bản này là những phần của công cụ phát triển Java (Java Development Kit), nó cũng có những lớp hỗ trợ tính năng mạng, những nghi thức chung của Internet, và những hàm giao tiếp với người dùng. Bởi vì những lớp thư viện này được viết trong Java, nó có tính khả chuyển giữa các môi trường cũng giống như tất cả các ứng dụng khác của Java.

❖ Java Virtual Machine (JVM)

- JVM đóng vai trò rất quan trọng để các ứng dụng Java có thể thực hiện được. Nó hoạt động như một máy tính ảo, cũng có bộ lệnh, cấu trúc dữ liệu, bộ nhớ ù... Khi ứng dựng Java thực hiện (sau khi dịch, các ứng dụng Java có phần mở rộng là class), JVM tiến hành phân mã trong class đó thành bộ lệnh của JVM rồi thực hiện giống như máy PC thao tác với các ứng dụng thông thường. Do đó các class sau khi dịch có thể được thực hiện trên bất kỳ hệ điều hành nào thông qua máy tính ảo JVM.
- Bởi vậy, các class sau khi dịch có thể được thực hiện trên bất kỳ hệ điều hành nào thông qua máy tính ảo JVM.
- Hiện tại, JVM được xây dựng cho hầu hết các hệ điều hành và hệ xử lý hiện có, điều này có nghĩa là các ứng dụng viết bằng Java có đầy đủ điều kiện để phát triển.
- Do phải hoạt động thông qua máy tính ảo JVM nên tốc độ thực hiện ứng dụng của Java khá chậm.
- Trên thị trường hiện nay có nhiều bộ công cụ lập trình cho Java : Java Workshop của Sun Microsystems, Visual J của Microsoft, Symantec Cafe của Symantec … Tất cả đều có điểm chung là hỗ trợ tối đa cho người lập trình. Ở luận án tốt nghiệp này chúng em đã sử dụng Symantec Cafe phiên bản 1.8 để viết chương trình này. Symantec Cafe 1.8 là một công cụ lập trình Java của hãng Symantec, tuy là một phiên bản cũ nhưng nó vẫn hỗ trợ tốt cho việc lập trình Java.

2. ***Các tính chất , ưu khuyết điểm của Java.***

a. Các tính chất cơ bản:

- Đơn giản(simple) :Cú pháp của java thực ra giống cú pháp của C++ trong các phiên bản mới đây. Mặc dù java không phải là ngôn ngữ được ưa chuộng nhất hiện nay nhưng trong thời điểm này java là ngôn ngữ hay hơn cả. Một điểm khác nữa là java rất nhỏ, nó có thể tạo ra những phần mềm chạy độc lập trên các máy tính nhỏ kích thước bộ biên dịch và các lớp thư viện của nó chỉ có 40K.

- Hướng đối tượng(Qbect oriented): mọi ứng dụng của java điều phải được xây dựng trên các đối tượng. mỗi lớp có nhiều phương thức(method) và vùng(field). Phương thức là những chức năng mà đối tượng dùng để trả lời các tác động lên đối tượng.

- Phân tán(distributed): Java được thiết kế để hỗ trợ các ứng dụng phân tán bằng các lớp mạng(java net). Ví dụ như lớp URL của java có thể truy xuất dễ dàng đến máy chủ ở xa, nó có thể mở hoặc truy cập đến các đối tượng thông qua mạng dễ dàng như một lập trình viên đang sử dụng ngay trên máy của mình.

- kiểu tỉnh(statically typed): tất cả các đối tượng được sử dụng trong chương trình phải được khai báo trước khi sử dụng. Điều này giúp cho trình biên dịch java có thể xác định và báo cáo các xung đột.

- Biên dịch(Compiled): trước khi chúng ta chạy một chương trình bằng java thì chương trình này phải được dịch lại bằng một trình dịch java(máy ảo java) kết quả sẽ đưa ra một file “bye-code” tương đương với một file mã máy, có thể thực hiện được trong bất kỳ hệ điều hành nào mà có trình thông dịch java(JVM). Trình thông dịch này đọc file “bye-code” và dịch các lệnh “bye-code” sang ngôn ngữ máy.

- Thông dịch(Interpreter): java là một ngôn ngữ thông dịch(thật ra nó là ngôn ngữ vừa thông dịch vừa biên dịch) nên nó trở nên khá khả chuyển. các ứng dụng java dễ dàng chạy trên các máy tính với các nền phần cứng khác nhau như Macintosh, Intel, Sun, Alpha,… chỉ cần đi kèm với các ứng dụng đó là trình thông dịch, bộ gỡ rối và nhất là bộ thư viện thời gian động(runtime library).

- Mở rộng(Extensible): các chương trình java hỗ trợ các hệ thống riêng mà các hàm được viết bằng một ngôn ngữ khác thường là C++,Visual Basic. Hỗ trợ cho các hệ thống riêng giúp cho người lập trình viết các hàm mà có thể thực hiện nhanh hơn các hàm tương đương viết bằng java. Các hệ thống riêng này được liên kết động với chương trình java. Khi java được cải thiện về tốc độ thì các hệ thống riêng này sẽ không cần thiết nữa.

- Mạnh mẽ(robust): giúp cho lập trình viên tạo nên những chương trình chắc chắn, không phạm nhiều vào những lỗi khi chạy(runtime error). Java không cho phép các lập trình viên khai báo các biến một cách tuỳ tiện mà các biến này phải tường minh. Ngay cả kiểu dãy cũng không phải là một con trỏ trong C mà là một kiểu thực. Nhờ đó mà những lỗi thường gặp như cấp phát bộ nhớ, bộ nhớ tràng, trùng lắp bộ nhớ,.. đã được java giải quyết triệt để.

- An Toàn(Secure): hệ thống java không gì kiểm tra mọi sự truy cập bộ nhớ mà còn đảm bảo không có virus nào làm ngưng một applet đang chạy.

- Bảo mật(Sercurity): Java được viết cho các ứng dụng chạy trên môi trường phân tán do đó java phải được thiết kết trên một hệ thống không virus, không bị phá rối, không biến con trỏ. Bộ thông dịch luôn kiểm tra chặt chẽ các mã byte.

- Khả chuyển: Do tạo được các máy ảo java tương thích với môi trường được cài đặt sẽ tạo nên kiến trúc trung tính trong java, các ứng dụng java viết sao cho chạy được trong máy ảo java. Mặt khác, các kiểu dữ liệu của java được định nghĩa không phụ thuộc vào bộ xử lý hay hệ điều hành mà các ứng dụng cài đặt.

- Hiệu quả cao: rõ ràng so với chương trình được biên dịch hoàn toàn bằng C hay C++ các chương trình java không có hiệu quả cao hơn. Nhưng với tính năng giao diện đồ hoạ, sự đơn giản, nhỏ gọn từ đó ta có thể nói là hiệu quả chung của các ứng dụng java là đáng khích lệ so với tính phức tạp của C/C++.

- Đa tuyến: Tính năng này cho phép chúng ta có thể thực hiện nhiều tiến trình song và tương hỗ lẫn nhau tránh được tính tuần tự nhàm chán và những thời gian chết trong chờ đợi.

b. Ưu điểm của java

- Được Sun mô tả như là một ngôn ngữ lập trình đơn giản, hướng đối tượng, hiểu - mạng, có thể biên dịch, mạnh, an toàn, độc lập với cấu trúc, dễ di chuyển, hiệu suất cao, đa luồng, và có tính động. Những điều đó không dễ giải thích. Vậy thì, cụ thể Java làm được gì?

- Về cơ bản, nó giúp các nhà phát triển phần mềm thực hiện được những việc sau:

♦Thứ nhất: họ có thể xây dựng nên các applet Java, đó là những trình ứng dụng mini được phân phối qua Internet và chạy trong một trình duyệt Web hiểu Java. Các applet Java tăng cường cho trang Web khả năng tương tác phong phú hơn và tính đa phương tiện tốt hơn so với khi dùng HTML bình thường. Applet hoạt động giống như cung cách bạn đặt một trang web với các siêu văn bản trên một server và một máy khách (client) có thể tải trang đó xuống theo yêu cầu để xem các văn bản đã sắp đặt theo khuôn dạng. Tương tự, bạn viết và biên dịch một chương trình applet Java và đặt một tham chiếu URL hoặc HTML tới nó trong trang web. Khi một client duyệt qua trang web này, mã nhị phân của applet Java được tải xuống client đó cùng các tệp văn bản và đồ hoạ. chương trình duyệt chứa một JVM và nó sẽ thực hiện applet trên máy tính của client.

♦Thứ hai: các nhà phát triển phần mềm có thể xây dựng các trình ứng dụng hoàn chỉnh bằng Java, như bộ xử lý văn bản, bảng tính, hoặc bộ chương trình văn phòng tổng hợp (như Corel đang làm chẳng hạn). Ưu điểm của cách làm này là các trình ứng dụng chỉ cần viết một lần mà chạy được trên hầu hết mọi loại máy tính.

♦Thứ ba: Java đáp ứng không những tính dễ chuyển mang mà còn cả cách xử lý đồng nhất của chương trình trên các hệ thống khác nhau. Đầu tiên mã nguồn Java được biên dịch để sinh ra mã đối tượng gọi là bytecode, bytecode không phải là mã nhị phân của bất kỳ máy tính đang tồn tại thực tế nào mà đó là một loại mã máy kiến tạo, Bạn sẽ thực hiện một chương trình Java bằng cách chạy một chương trình khác gọi là Java Virtual Machine hay là JVM, JVM đọc chương trình bằng bytecode và thông dịch hoặc biên dịch nó ra theo hệ lệnh thực tế, JVM biến tất cả mọi nền phần cứng và phần mềm trở nên giống nhau dưới con mắt của chương trình Java. Chạy bytecode trên một JVM là lý do vì sao các phần mềm Java là "viết một lần, chạy khắp nơi"

♦Thứ tư: Việc quản lý bộ nhớ: So với ngôn ngữ C v à C^{++}, Các chương trình Java được quản lý về bộ nhớ ở mức hệ thống và người lập trình không bao giờ phải lo lắng về chuyện đó. Thư viện thời gian chạy của Java sẽ giám sát các cấu trúc dữ liệu. Khi không còn một tham chiếu nào tới một cấu trúc dữ liệu thì nó không thể là đang được sử dụng vì chương trình không có cách gì để đọc hoặc ghi nó. Lúc đó nó sẽ là đối tượng của việc dọn dẹp bộ nhớ.

Java hướng tới việc dọn dẹp bộ nhớ tự động. Việc dọn dẹp bộ nhớ tự động ảnh hưởng tới tính năng vì nó liên quan đến các quá trình khác chạy trong nền sau để giám sát việc sử dụng bộ nhớ. Tuy nhiên thực tế đã chỉ ra rằng, ở đây cũng có một sự cân nhắc đáng giá. Một hệ thống nhỏ thực hiện dọn dẹp bộ nhớ tự động đã dẫn tới sự cải thiện rất lớn thông qua việc gỡ bỏ một loạt các lỗi của các chương trình Java. Có thể so sánh, các chương trình C++ chịu trách nhiệm quản lý các đống (heap) bộ nhớ của riêng chúng do vậy chúng phải có mã dài hơn, mất nhiều thời gian gỡ rối hơn và các chương trình lớn thường dẫn tới các lỗi rất khó phát hiện và xử lý về việc dọn dẹp bộ nhớ.

- Java là một môi trường độc lập, đó là một lợi thế quan trọng cho phép Java hơn hẳn những ngôn ngữ khác, đặc biệt là cho những hệ thống cần làm việc trên nhiều môi trường khác nhau, Java là một môi trường độc lập ở cả trên nền hệ thống lẫn dưới mức thấp như hệ xử lý nhị phân. Nó có khả năng chuyển từ hệ thống máy tính này sang hệ thống máy tính khác không phụ thuộc vào cấu trúc của máy hay hệ điều hành hoạt động trên máy.

c. Nhược điểm của Java

- Java có tốc độ thực thi chương trình phải thông qua JVM nên tốc độ rất chậm so với các ngôn ngữ khác.

- khi cài đặt và tiết lập các ứng dụng java thường rất phức tạp hơn các ngôn ngữ khác vì thế mà java hiện nay chưa được các lập trình viên ưa chuộng.

II.Một số kỹ thuật Lập trình mạng trong java

1. *Các kiến thức cơ bản về Networking*

- Các máy tính chạy trên mạng Internet truyền thông với nhau dùng các Protocol TCP, UDP. Mô hình mạng 4 lớp được mô tả bằng hình vẽ dưới đây:

Application (HTTP,ftp,telnet)
Transport (TCP/IP,UDP)
Network (IP,.)
Link (device driver)

- Khi bạn viết các chương trình Java có truyền thông qua mạng, điều này có nghĩa là bạn đang lập trình ở lớp application. Nhìn chung, bạn không cần quan tâm tới các protocol TCP và UDP. Thay vì vậy, bạn có thể dùng các lớp trong package java.net. Các lớp này cung cấp việc truyền thông qua mạng độc lập hệ thống. Tuy nhiên, bạn cần hiểu rõ sự khác biệt giữa TCP và UDP để xác định rõ những lớp nào trong thư viện Java mà bạn sẽ sử dụng.

- Khi hai chương trình muốn truyền dữ liệu cho nhau một cách đáng tin cậy, chúng thiết lập một connection và gửi data qua lại thông qua connection đó. Điều này giống như một cuộc điện thoại--nếu bạn muốn nói chuyện với tôi ở HCM City, một connection được thiết lập khi bạn quay số của tôi và tôi trả

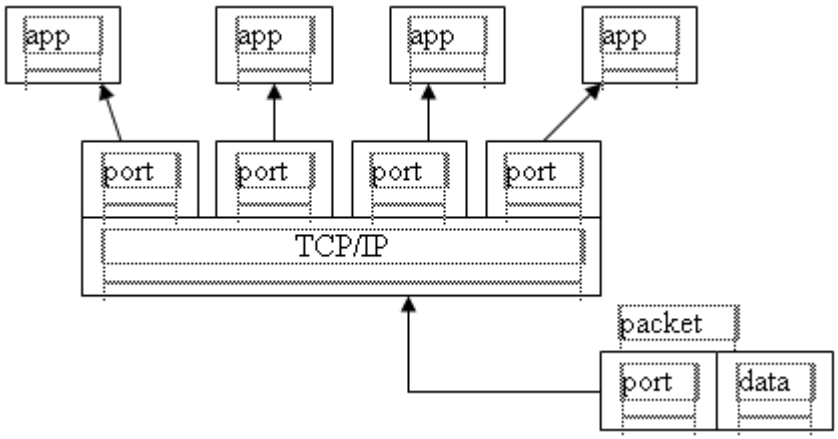
lời. Bạn gửi data qua lại thông qua connection bằng cách nói với người kia thông qua đường điện thoại. Giống như các công ty điện thoại, TCP đảm bảo rằng data được gửi từ một đầu connection tới đầu kia không mất mát và đúng thứ tự (nếu không, một lỗi sẽ được thông báo).

2. *Ports:*

- Nói một cách tổng quát, một máy tính nối mạng là một connection vật lý đối với mạng đó. Tất cả dữ liệu gửi cho một máy tính thông qua conection đó. Tuy nhiên, dữ liệu có thể được gửi cho những ứng dụng khác nhau trên máy đó. Vậy thì làm cách nào máy tính biết ứng dụng nào sẽ nhận dữ liệu được gửi đến? Điều này được giải quyết thông qua việc sử dụng Ports, mỗi ứng dụng mạng có một port tương ứng.

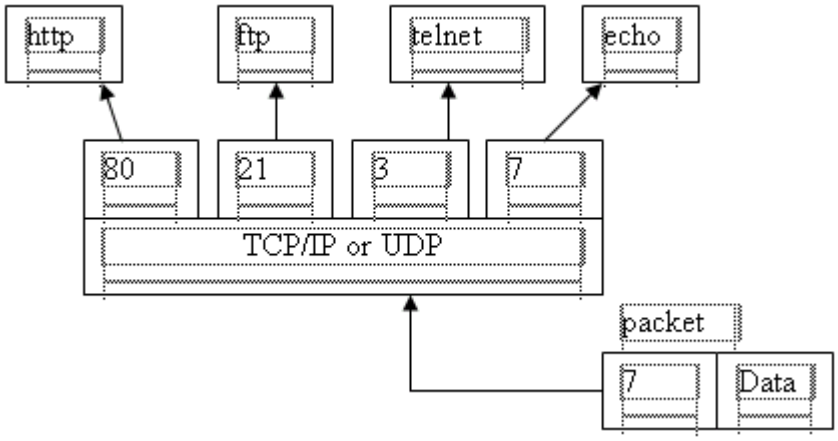
- Dữ liệu truyền qua mạng có kèm theo thông tin địa chỉ nhằm xác định máy tính và port đích. Mỗi máy tính được xác định bằng một địa chỉ IP 32-bits, IP protocol dùng địa chỉ này để phân phát dữ liệu đúng cho từng máy. Port được xác định bằng một số 16-bits, các protocol TCP và UDP dùng port number để phân phát data tới đúng cho từng ứng dụng.

- Trong việc truyền nhận data dựa trên connection, một ứng dụng thiết lập một connection với một ứng dụng khác bằng cách gắn một socket cho mỗi port number. Điều này có ý nghĩa đăng kỳ ứng dụng với hệ thống để ứng dụng có thể nhận tất cả data được gửi đến cho port đó. Không thể có hai ứng dụng dùng chung một port. Mọi cố gắng gắn một socket với một port đã dùng đều sẽ thất bại. Trong việc truyền nhận data dựa trên datagram, datagram chứa port number của ứng dụng đích mà nó gửi tới.



- Định nghĩa: Các protocol TCP và UDP dùng ports để map incoming data cho một quá trình đang chạy trên một máy tính.

- Port number nằm trong khoảng 0-65535 (vì ports được biểu diễn bằng số nguyên 16-bits). Những port nằm trong khoảng 0-1023 là những port dành riêng cho những dịch vụ quen thuộc như HTTP, FTP và các dịch vụ của hệ thống (những port này gọi là các well-known port). Những ứng dụng mạng của bạn không nên dùng những port trong khoảng này.



- Thông qua những lớp trong package java.net, những chương trình Java có thể dùng TCP hay UDP để truyền nhận data qua Internet. Các lớp URL, URLConnection, Socket và ServerSocket dùng TCP. Các lớp DatagramPacket và DatagramServer dùng UDP.

- Java sử dụng HTTP để phân phát các Applet đa nền, có thể chạy trong môi trường Browser. Nhìn chung, đây là công dụng chính của Java: tạo ra các trang HTML có nội dung động. Tuy nhiên, đây chỉ là mặt ngoài của cái mà Java có thể làm được thực sự. Các Net-package và kiến trúc Java cho phép nó được dùng như một kiến trúc động, có thể chủ động dùng nguồn code, data, và input thông qua Internet. Bằng cách tổng hợp các Java-package, programmers có thể phối hợp chương trình của mình cùng các protocol Telnet, FTP, NNTP, WWW để tạo ra các ứng dụng mạng, thay vì chỉ chạy trên một máy như trước đây.

- Java tương tác với Internet theo cách riêng của nó, dữ liệu kéo về dưới dạng các file bytecode (.class), các file khác như ảnh, audio hay input từ việc tương tác với các user khác. Chức năng này được giao tiếp chính thông qua môi trường Browser support Java; mặc dù vậy, interpreter cũng có thể sử dụng các connection mạng. Nhằm đáp ứng hai khả năng quan trọng là tính có thể mở rộng và tính đa nền, Java đã cung cấp một kiến trúc hướng đối tượng không bị ràng buộc bởi việc hiện thực chương trình được compile từ trước khi thực thi (dạng file.EXE). Ngoài ra, để đáp ứng được những yêu cầu của người sử dụng, Java phải đảm bảo tính an toàn, hiệu suất cao. Tuy nhiên, cho đến thời điểm hiện tại, tốc độ thực thi một chương trình Java còn quá chậm. Hy vọng điều này sẽ được cải thiện hơn khi có những phần cứng support Java riêng biệt.

- Các tính chất an toàn, đa nền,...của ngôn ngữ Java được giải quyết bằng interpreted design. Bằng cách Compile code ra dạng máy ảo, và tạo ra memory layout tại thời điểm chạy chương trình thay vì tại thời điểm compile, Java có khả năng truy xuất điều kiện của code trước khi nó được thực thi trên một máy client.

Tất cả những ưu điểm trên phải trả giá cho hiệu suất thực thi chương trình thấp. Tuy nhiên, điều này đã được khắc phục phần nào bằng cách tạo code trung gian dưới dạng bytecode, cung cấp khả năng thực hiện chương trình Multithread khá dễ dàng, và đặc biệt hơn là chiến lược quản lý bộ nhớ với việc dọn rác tự động.

3. *Networking:*

- Việc sử dụng những khả năng networking do Java support khá dễ dàng so với C và C++. Applet được nhúng trong các file HTML. Để chạy những Applet qua mạng, việc trước tiên cần là là load các trang HTML này về máy cục bộ. Các applet được nhúng trong các trang HTML thông qua phần khai báo APPLET. Ngoài những lớp được hiện thực bởi applet, những lớp thư viện khác do Browser cung cấp.

- Việc load các file ảnh và file audio được thực hiện thông qua lớp URL (package java.net). Lớp này biểu hiện một Uniform Resource Locator, là địa chỉ của tài nguyên nào đó trên mạng. Lấy ví dụ, để load một file ảnh từ mạng, chương trình Java đầu tiên cần tạo một URL chứa địa chỉ chỉ tới file ảnh đó, sau đó dùng một số hàm cần thiết để connect và truy xuất file ảnh đó.

- Nhìn chung, điểm nổi bật của Networking do Java mang lại là tính tiện lợi và dễ sử dụng. Điều này có thể hiểu rõ hơn trong phần giới thiệu về URLs, Socket ở những phần sau của tài liệu này.

- Giao tiếp giữa phần hiện thực client và Browser ở máy local:

4. *URLs*

- Nếu bạn từng giao tiếp với World Wide Web, hẳn bạn không lạ gì với khái niệm URL và chắc bạn cũng đã dùng các URL để truy xuất các trang HTML trên Web. URL là từ viết tắt của Uniform Resource Locator, là một tham chiếu (một địa chỉ) tới một tài nguyên trên Internet.

- Nếu bạn chưa quen, thì tốt hơn hãy xem (dù không chính xác hoàn toàn) URL là tên của một file trên mạng vì hầu hết các URL tham chiếu tới 1 file trên máy nào đó trên mạng. Tuy nhiên, bạn nên nhớ rằng URL có thể tham chiếu tới bất kỳ tài nguyên mạng nào, chẳng hạn như database queries và command output.

Có 5 loại URL : file, http, gopher, news, partials.

- Cú pháp chung của một URL là :

<protocol>://<địa chỉ máy>:<port>/<đường dẫn resource>/<tên resource>

File URL :Áp dụng cú pháp chung trên thì file URL chỉ tới file foobar.txt trong thư mục pub/files/ trên máy file Server tên là "ftp.yoyodyne.com":

file://ftp.yoyodyne.com/pub/files/foobar.txt

tương tự ta có file URL chỉ tới thư mục "pub" trên máy FTP Server :

file://ftp.yoyodyne.com/pub

- Http URL :

Ví dụ :

http://www.yoyodyne.com/pub/files/foobar.html

http://www.yoyodyne.com:1234/pub/files/foobar.html

- Gopher URL :

Ví dụ :

gopher://gopher.yoyodyne.com/

gopher://gopher.banzai.edu:1234/

- News URL :

Không giống như những URL khác, để chỉ tới 1 newsgroup tên là "rec.gardening" thì news URL là:

news:rec.gardening

- Partials URL :

Đây là một loại URL dùng để chỉ tới một resource có cùng thư mục, cùng tên máy với một resource đã có sẵn. Ví dụ : http URL chỉ tới 1 file như sau :

http://www.yoyodyne.com/pub/afile.html

Vậy thì lúc này chúng ta có thể dùng *partial URL*, hay *relative* URL để chỉ tới một file khác trong cùng một thư mục, cùng một máy với file tên là "afile.html" như trên. Ví dụ: trong thư mục trên có file tên là "anotherfile.html", thì lúc này partial URL để chỉ tới file trên là :anotherfile.html

- Tất cả các URL đều có 2 phần tử chính:
  - Kiểu Protocol.
  - Tên tài nguyên.
- Cách dễ nhất để tạo một URL là dùng một chuỗi làm đối số cho URL cconstructor:

URL u = new URL("http://java.sun.com");

- Đây là một URL tuyệt đối vì nó đặc tả toàn bộ tên tài nguyên. Một constructor hữu dụng khác là một URL tương đối:

URL data = new URL(u,"conference/conference.html");

- URL này đặc tả file conference.html, nằm trong thư mục conference của URL u.

Trong chương trình, URL được sử dụng để load file chứa các nhóm hiện đang tồn tại, file chứa danh sách các user trong một nhóm, log file (file chứa thời điểm các user login, logout; các nhóm được tạo ra, delete), file help. Để phát triển đề tài này thành các đề tài khác yêu cầu thêm một số chức năng như truyền nhận file, truy xuất cơ sở dữ liệu và cập nhật ngay ở client khi dữ liệu ở server có thay đổi, cần phải có những kiến thức thêm nữa về cách sử dụng URL trong Java. Tài liệu này có một phần phụ lục ở phía sau về cách dùng URL trong Java. Nếu quan tâm, bạn có thể tham khảo.

5. Applet Context

- Một applet chạy trong một browser như Netscape hay Apllet viewer. Một applet có thể yêu cầu browser làm việc gì đó cho nó, ví dụ như, fetch một file audio, show một trang Web khác,... Tuỳ vào loại browser mà có thể thực hiện yêu cầu này hoặc không. Để liên kết với browser, applet gọi hàm getAppletContext. Hàm này trả về một đối tượng kiểu AppletContext
- Một số hàm liên quan đến việc Applet yêu cầu hiển thị một URL bên trong browser:
  - *public AppletContext getAppletContext()*

    - Tạo ra một handle (context của applet) trên môi trường browser của applet. Trong hầu hết các browser, ta có thể dùng thông tin này để điều khiển browser mà applet đang chạy trên đó.
  - *public abstract void showDocument(URL url, String target)*

    - Yêu cầu browser chứa applet show trang web xác định bởi thông số url. Thông số target chỉ ra nơi hiển thị trang web (... để xem thêm các giá trị có thể có của target trong thư viện API của Java).

6. Socket

Minh họa cơ chế Socket

- Trong mô hình mạng để hai máy tính có thể trao đổi thông tin cho nhau thì cần phải tạo ra kết nối giữa chúng. Trong quá trình làm việc người ta nhận thấy rằng những nhà lập trình ứng dụng rất khó khăn trong việc thiết lập kết nối và truyền tải dữ liệu giữa các máy tính với nhau. Vì thế người ta xây dựng khái niệm socket, khái niệm này được đưa ra đầu những năm 80 bởi các nhà khoa học máy tính ở California tại Berkeley. Khái niệm này được đưa ra từ ý tưởng phân tầng, trong đó Windows sockets Application Programming Interface(Winsock API) là thư viện các hàm do hãng Berkeley Software Distribution of UNIX đưa ra. Nhằm làm đơn giản hoá quá trình thiết lập kết nối và chuyển dữ liệu. Socket dựa trên giao thức TCP/IP tạo môi trường trung gian cho các ứng dụng và giao thức bên dưới.

- Socket được xem là một cấu trúc dữ liệu trừu tượng(asbtraction data structure) dùng tạo ra một kênh truyền(channel) để gởi và nhận dữ liệu giữa các process trong cùng chương trình hay giữa các máy trong cùng môi trường mạng với nhau. Hay nói một cách đơn giản hơn chúng ta xem socket như là "cơ chế ổ cắm". Khi kết nối giữa client và Server tương tự như việc cắm phích điện vào ổ cắm điện. client thường được xem như là phích cắm điện, còn server được xem như là ổ cắm điện, một ổ cắm có thể cắm vào đó nhiều phích điện khác nhau cũng như một server có thể phục vụ cho nhiều client khác nhau.

- Ta sử dụng các URLs và URLConnections để truyền thông qua mạng ở cấp cao và dành cho một mục đích đặc biệt:truy xuất tài nguyên trên Internet. Đôi khi chương trình của ta đòi hỏi việc truyền thông qua mạng ở mức thấp hơn, ví dụ như khi ta viết một ứng dụng Client/Server (cụ thể là phần mềm này đã thực hiện điều đó).

- Trong những ứng dụng client/server, server cung cấp một dịch vụ nào đó, chẳng hạn như xử lý các database queries, gửi giá cả chứng khoán hiện tại,...Client dùng dịch vụ do server cung cấp để hiển thị kết quả cho User, tạo ra những chỉ dẫn cần thiết cho người đầu tư,... Việc truyền dữ liệu giữa client và server phải đáng tin cậy, không bị mất và không được tới sai thứ tự do server gửi.

- TCP cung cấp kênh truyền đáng tin, point-to-point, mà những ứng dụng client/server trên Internet sử dụng. Những lớp Socket và ServerSocket trong java.net package hỗ trợ cho việc thực hiện các kênh truyền TCP độc lập hệ thống.

- Một Socket là một end-point của một liên kết giữa hai chương trình chạy trên mạng. Các lớp socket được sử dụng để thể hiện connection giữa một chương trình client và một chương trình Server. Package java.net cung cấp hai lớp: Socket và ServerSocket tương ứng với client và server.

- Trong quá trình truyền, nhận dữ liệu cần có một máy đóng vai trò là server và một máy đóng vai trò client, đầu tiên server phải tạo ra một socket và chờ đợi các yêu cầu kết nối từ client. client tạo ra socket cho riêng nó xác định vị trí server (dựa vào tên của server hay địa chỉ của server trong mạng) và tiến hành việc kết nối với server, sau khi kết nối được thiết lập client và server có thể tiến hành việc trao đổi dữ liệu với nhau.

- Chương trình Server thường listen ở một port riêng biệt, đợi các connection request từ các chương trình client. Khi có một connection request, client và server thiết lập một connection mà qua đó chúng sẽ trao đổi dữ liệu với nhau. Suốt quá trình connecton, client được gán cho một port number cục bộ, và bind một socket cho nó. Client truyền (nhận) dữ liệu cho server bằng các ghi (đọc) socket. Tương tự, server nhận một port number local mới (nó cần một port number mới để nó có thể tiếp tục listen các connetion request khác ở port ban đầu). Server cũng bind một socket cho port cục bộ của nó và liên lạc với client tương ứng thông qua socket này.

a. Sử dụng Socket ở Client

- Dưới đây là một chương trình ngắn minh hoạ việc sử dụng Socket ở Client. Trong ví dụ, client thiết lập một connection với Echo server (port = 7), ở máy ResearchCC dùng lớp Socket trong thư viện API của Java. Client nhập một hàng từ standard input stream, gửi cho Echo server. Echo server nhận được, gửi trả hàng này về lại client. Client đọc hàng này và xuất lại ra màn hình.

- Việc sử dụng Socket trong chương trình Client của phần mềm này có đôi chút phức tạp hơn nhưng những ý tưởng chính là hoàn toàn giống nhau.

```
import java.io.*;
import java.net.*;
public class EchoTest {
   public static void main(String[] args) {
      Socket echoSocket = null;
      DataOutputStream os = null;
      DataInputStream is = null;
      DataInputStream stdIn = new DataInputStream(System.in);
      try {
         echoSocket = new Socket("ResearchCC", 7);
         os = new DataOutputStream(echoSocket.getOutputStream());
         is = new DataInputStream(echoSocket.getInputStream());
      } catch (UnknownHostException e) {
         System.err.println("Don't know about host: ResearchCC");
      } catch (IOException e) {
                  System.err.println("Couldn't get I/O for the connection to:
ResearchCC");
      }
      if (echoSocket != null && os != null && is != null) {
         try {
            String userInput;
            while ((userInput = stdIn.readLine()) != null) {
```

```
            os.writeBytes(userInput);
            os.writeByte('\n');
            System.out.println("echo: " + is.readLine());
          }
          os.close();
          is.close();
          echoSocket.close();
        } catch (IOException e) {
          System.err.println("I/O failed on the connection to: ResearchCC");
        }
      }  }
}
```

Giải thích chương trình trên:

- Ba hàng sau trong khối try của phương thức main() buộc phải có. Chúng thiết lập socket connetion giữa client-server và mở input, outputstream trên socket tạo ra:

```
echoSocket = new Socket("ResearchCC", 7);
os = new DataOutputStream(echoSocket.getOutputStream());
is = new DataInputStream(echoSocket.getInputStream());
```

Hàng đầu tạo ra một đối tượng Socket, đặt tên là echoSocket. Socket constructor được sử dụng ở đây yêu cầu tên máy và port number mà ta muốn connect tới. Chương trình ví dụ dùng host name ResearchCC. Đối số thứ hai là port number. Port 7 là port mà Echo server listen.

Hàng thứ hai và thứ ba mở một output stream và một input stream trên socket vừa thiết lập. EchoTest đơn thuần chỉ cần write tới output stream và read từ input stream để truyền dữ liệu với server thông qua socket. Nếu bạn chưa quen với các stream trong Java, bạn có thể xem thêm phần các Stream cũng trong đề tài này.

- Phần kế đọc từ standard input stream của EchoTest một hàng mỗi lần. EchoTest write ngay input text (theo sau bởi một newline character) tới output stream:

```
String userInput;
while ((userInput = stdIn.readLine()) != null) {
os.writeBytes(userInput);
os.writeByte('\n');
System.out.println("echo: " + is.readLine());
}
```

- Hàng cuối trong vòng lặp while đọc một hàng từ input stream. Phương thức readLine() block cho tới khi server echo thông tin trở về cho EchoTest. Khi readLine() return, EchoTest in hàng thông tin ra standard output. Vòng lặp while tiếp tục-- EchoTest đọc input từ user, gửi nó cho Echo server, nhận trả lời từ server, và hiển thị nó-- cho tới khi user đánh vào một end-of-input character.

- Khi user đánh vào một end-of-input character, vòng lặp while kết thúc, chương trình tiếp tục thực thi ba hàng kế:

```
os.close();
is.close();
echoSocket.close();
```

- Ba hàng code này đóng các input, output stream, rồi đóng socket connection tới server. Thứ tự thực hiện ở đây rất quan trọng, ta nên đóng các stream kết nối với một socket trước khi đóng chính socket đó.

- Nhìn chung, các bước cần tiến hành để dùng socket ở phía client như sau:

1. Mở một socket.
2. Mở input stream và outputstream ứng với socket đó.
3. Read và write tới stream tuỳ thuộc vào nghi thức của server
4. Đóng các stream.
5. Đóng socket.

- Chỉ có bước ba là khác nhau giữa các client, do dựa vào server. Các bước còn lại hầu như giống nhau.

b. Sử dụng Socket ở Server

- Chương trình Server bắt đầu bằng việc tạo ra một instance của lớp ServerSocket để listen một port được đặc tả. Khi hiện thực một chương trình Server, ta nên chọn một port mà không được dành sẵn cho các dịch vụ khác:

```
try {
   serverSocket = new ServerSocket(4444);
} catch (IOException e) {
    System.out.println("Could not listen on port: " + 4444 + ", " + e);
    System.exit(1);
 }
```

- Bước kế tiếp, Server accept một connection request từ một client:

```
Socket clientSocket = null;
try {
     clientSocket = serverSocket.accept();
} catch (IOException e) {
      System.out.println("Accept failed: " + 4444 + ", " + e);
       System.exit(1);
}
```

- Sau khi sinh ra socket ứng với client yêu cầu connect,server dựa vào socket và các input, output stream ứng với socket này để thực hiện việc read, write dữ liệu. Điều này hoàn toàn giống như cách thức đã mô tả ở phần sử dụng soclet ở Client được mô tả ở phần trên. Cụ thể là các việc sau:

1. Mở một input và output stream ứng với socket.
2. Read và write tới socket.

- Trong ví dụ, có nhiều client đồng thời đưa ra các connection request với server tại port mà server listen (port 4444). Có hai cách để giải quyết vấn đề này:

1. Các connection request được xếp hàng, và Server phải accept các connection tuần tự
2. Phục vụ các connection request đồng thời bằng việc dùng các thread-- Mỗi thread xử lý một connection

Ví dụ đã chọn cách thứ hai để hiện thực vì nó hợp lý hơn: mọi user cần phải được xử lý bình đẳng như nhau.

Giải thuật cho phần này như sau:

while (true) {
accept a connection;
create a thread to deal with the client;
end while

Thread được sinh ra sẽ read và write tới connection ứng với connection đó khi cần thiết.

c. Thư viện các hàm socket (API) trong Java.

- Trong Java người ta cũng xây dựng các lớp về socket phục vụ cho việc truyền tải dữ liệu dễ dàng và nhanh chóng, các lớp này được đóng gói trong gói Java.net. Một số lớp cần thiết trong gói Java.net

♦ Lớp InetAddress

- Vì địa chỉ Internet theo số IP và theo tên rất thường dùng khi kết nối vào mạng cho nên Java xây dựng hẳn một lớp InetAddress dành riêng cho việc quản lý địa chỉ theo tên và số lớp. Lớp InetAddress cung cấp các phương thức static thông dụng nhất dùng để chuyển đổi và truy xuất địa chỉ IP (không có phương thức khởi dựng cho lớp này). Thường ta sẽ quan tâm đến các phương thức sau:

✓ *public static InetAddress getLocalHost() throws UnknownHostExceptiongetByName*

Trả về đối tượng InetAddress là địa chỉ máy cục bộ(local host).

✓ *public static InetAddress getByName(String host) throws UnknownHostException*

Phương thức này nhận địa chỉ của một máy bằng kiểu chuỗi và trả về đối tượng InetAddress thay mặt cho địa chỉ máy này.

✓ *public static InetAddress[] getAllByName(String host) throws UnknownHostException*

Phương thức này nhận địa chỉ của một máy bằng kiểu chuỗi và trả về tất cả đối tượng InetAddress thay mặt cho địa chỉ máy này.

✓ *public byte[] getAddress()*

Trả về địa chỉ IP của đối tượng InetAddress dưới dạng một dãy các byte. Vị trí byte cao nhất nằm ở byte 0.

✓ *public String getHostAddress()*

Trả về địa chỉ IP của đối tượng InetAddress dưới dạng một chuỗi được định dạng phân thành làm 4 nhóm %d.%d.%d.%d (ví dụ "127.16.11.12").

♦Lớp Socket

- Lớp Socket dùng tạo kết nối từ phía máy khách với máy chủ trương được khởi dựng bằng các phương thức sau:

✓ *public Socket(String host, int port) Throws UnknownHostException, IOException*

Hàm constructor. Tạo ra một stream socket và connect nó với port được đặc tả bởi thông số port, trên host đặc tả bởi thông số host. Ngầm định là tạo ra stream socket (ngoài ra có thể tạo ra datagram socket nếu đặc tả thêm thông số). Trong chương trình, thông số

InetAddress được lấy bằng việc gọi hàm getHost() sau khi có chuỗi URL chứa chương trình Client.

✓ *public Socket(InetAddress address, int port) Throws IOException*

Tạo ra một Socket kết nối từ địa chỉ là đối tượng InetAddress và số cổng port.

✓ *public Socket(String host, int port, boolean stream) throws IOException.*

Tạo ra một socket kết nối theo địa chỉ host và số cổng port, tham số stream cuối cùng để quy định kết nối theo TCP(stream=true)hayUDP(stream=false). Tuy nhiên nếu áp dụng để tạo socket cho giao thức UDP nên sử dụng lớp thay thế là DatagramSocket.

- Các phương thức khác hỗ trợ cho lớp Socket từ phía máy khách bao gồm:

✓ *public InputStream getInputStream() Throws IOException*

Trả về một input stream thực hiện việc đọc dữ liệu từ socket này.

✓ *public OutputStream getOutputStream() throws IOException*

Trả về một output stream thực hiện việc ghi dữ liệu tới socket này.

✓ *public InetAddress getInetAddress()*

Trả về remote IP address mà socket này connect với. Từ trị trả về này, có thể gọi hàm getHostName từ lớp InetAddress để lấy hostName tương ứng. Hàm này được gọi trong chương trình khi server cần lấy hostName của client connect với nó.

✓ *public Int getPort()*

Lấy về số cổng dùng kết nối của máy chủ.

✓ *Synchronized void close() throws IOException*

Đóng kết nối lại.

♦ ***Lớp ServerSocket***

- Lớp ServerSocket dùng tạo kết nối máy chủ với máy khách. Đối tượng ServerSocket được tạo ra trên máy chủ và lắng nghe những kết nối từ phía máy khách gửi đến theo một số cổng định trước. Đối tượng ServerSocket được khởi dựng từ phương thức sau:

✓ *public ServerSocket(int port) throws IOException*

Port là số hiệu cổng mà đối tượng ServerSocket phải lắng nghe để nhận biết những kết nối từ phía máy khách gởi đến. Nếu port = 0 thì tạo ra một server socket trên bất kỳ port nào trống.Chiều dài hàng đợi lớn nhất cho các yếu cầu connection là 50. Nếu một yêu cầu connection đến trong khi hàng đợi đầy, thì yêu cầu đó sẽ bị từ chối.

- Để chờ đợi kết nối từ các máy khác gửi đến đối tượng ServerSocket thường nhờ đến phương thức accept như sau:

✓ *Socket accept() throws IOException*

Phương thức này thực sự dừng lại chờ đợi cho đến khi nhận được thông tin kết nối sẽ trả về đối tượng socket của máy khách nơi có yêu cầu nối vào máy chủ. Phương thức này bị block cho tới khi connection được thực hiện.

- Cuối cùng máy chủ có thể đóng mọi kết nối bằng cách gọi phương thức close của đối tượng serversocket:

✓ *public void close() throws IOException*

♦ ***Lớp DatagramSocket***

Lớp này được dùng để chuyển đi một gói dữ liệu (biểu diễn bằng đối tượng DatagramPackage) theo giao thức UDP. Dữ liệu được gửi đi không an toàn có thể bị lỗi trên đường truyền. Dưới đây là một số phương thức thường dùng của lớp DatagramSocket.

- Phương thức khởi dựng để tạo kết nối UDP.

✓ *public DatagramSocket() throws SocketException*

- Phương thức khởi dựng để tạo kết nối UDP với số hiệu cổng port.

✓ *public DatagramSocket(int port) throws SocketException*

- Gói dữ liệu đi.

✓ *public void synchronized send(DatagramPackage p) throws IOException*

- Nhận gói dữ liệu về.

✓ *public void synchronized receive(DatagramPackage p) throws IOException*

- Đóng kết nối.

✓ *public void synchronize close();*

♦ ***Lớp DatagramPackage***

Lớp này dùng cho một gói chứa dữ liệu gửi đi trên mạng theo kết nối DatagramSocket. Một gói có thể chứa thông tin như chiều dài gói, các địa chỉ IP và số cổng mà từ đó gói dữ liệu được chuyển đi. Dưới đây là một số phương thức hữu dụng của lớp DatagramPackage.

- Phương thức khởi dựng gói có dữ liệu chứa trong bộ đệm buff[], chiều dài gói là len.

✓ *public DatagramPackage(byte buf[], int len)*

- Phương thức khởi dựng gói có dữ liệu chứa trong bộ đệm buff[], chiều dài gói là len, địa chỉ máy đích, và số hiệu cổng.

✓ *public DatagramPackage(byte buf[], int len, InetAddress iaddr, int port)*

- Trả về địa chỉ IP chứa trong gói dữ liệu

✓ *public InetAddress getAddress()*

- Trả về dữ liệu thật sự chứa trong gói.

✓ *public byte[] getData()*

- Trả về kích thước hay chiều dài gói dữ liệu.

✓ *public int getLength()*

- Trả về số hiệu cổng chứa trong gói dữ liệu.

✓ *public int getPort()*

♦ ***Các Stream:***

✓ *Class BufferedInputStream là một input stream.*

Public BufferedInputStream (InputStream in) : Hàm constructor. Tạo ra một input stream có đệm để đọc data từ input stream được khai báo.

Kích thước beffer ngầm định là 512-byte. Ta có thể khi báo kích thước buffer bằng constructor khác.

✓ *Class DataInputStream :* Ứng dụng sử dụng một data input stream để đọc các kiểu dữ kiệu nguyên thuỷ của Java từ một input stream lớp dưới, với đặc tính độc lập máy. Ứng dụng dùng data out put stream để ghi data mà sau này sẽ được đọc bởi một data input stream.

public DataInputStream(InputStream in) : Hàm constructor. Tạo ra một data input stream để đọc data từ input stream được khai báo. Trong chương trình, sử dụng thông InputStream là một BufferedInputStream được sinh ra từ input stream nhận được từ socket tương ứng.

public final String readLine() throws IOException : Đọc hàng text kế từ data input stream gọi nó. Phương thức này đọc thành công các bytes từ input stream lớp dưới cho đến khi hết một hàng. Kỳ hiệu chấm dứt dòng được xác định bằng các kỳ tự sau: kỳ tự CR ('\r'), kỳ tự newline ('\n'), một kỳ tự CR theo sau bởi một kỳ tự newline, hay kết thúc của stream input. Phương thức này bị block khi xảy ra một trong ba tình huống sau: một kỳ tự newline được đọc, một kỳ tự CR và byte đi liền sau nó được đọc (để xem có phải là kỳ tự newline hay không), dò thấy dấu hiệu chấm dứt một stream, hay mot □ IOException được sinh ra.

✓ Các kiến thức về output stream, buffered output stream cũng tương tự như input stream.

✓ *Class PrintStream :* Một print stream hiện thực một output stream filter, cung cấp các phương thức tiện lợi cho việc print các kiểu dữ liệu khác nhau.

public PrintStream(OutputStream out) : Xây dựng một print stream mà sẽ viết output của nó tới một output stream lớp dưới được đặc tả.

public void println(String s). Print một chuỗi tới output stream lớp dưới của Print Stream gọi hàm này.

7. ***Java Security***

- Các Java-application không thực hiện tính bảo mật như Java-applet. Với một Java-application, nó có thể đọc và ghi file, giao tiếp với thiết bị, connect với các socket,...Nhưng với Java-applet thì không như vậy. Có nhiều việc mà Java-applet không được phép làm, và nhiều tài nguyên mà Java-applet chỉ nên hạn chế truy xuất.

- Các applet có thể quan hệ tới mô hình Client/Server cổ điển theo cách: Web server là server của applet. Nó gửi applet tới máy client. Máy client là máy trên đó client chạy thực sự. Điều đó có nghĩa là khi ta browse một trang HTML có nhúng applet, máy của ta là client. Điều này làm rõ việc xác định những việc mà applet không được phép làm.

- Nếu một applet được load qua mạng, nó không được phép:

✓ Đọc, ghi, xoá, đổi tên file, tạo thư mục, liệt kê nội dung thư mục, kiểm tra sự tồn tại của file,... trên client file system.

✓ Khai báo bất kỳ hàm điều khiển mạng nào, định nghĩa các lớp trên client file system, thiết lập connection với một chương trình chạy trên một máy khác máy chứa applet đó.

Những điều trên dẫn đến hạn chế của chương trình: chỉ có thể đặt Server cùng một máy với Client, và cũng là máy chạy Web server.

8. Xử Lý Đa Tiến Trình(multitasking) và Đa Luồng(multithreading)

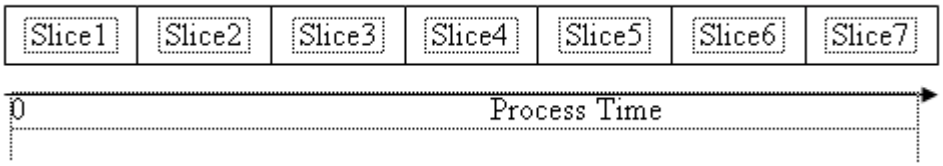
a. Đa Tiến Trình (multitasking)

- Lúc ban đầu, lập trình viên không có hệ điều hành mà giao tiếp trực tiếp với máy tính. Sao đó, lập trình viên xử lý tập tin batch đơn giản trên hệ điều hành hỗ trợ thực thi những tác vụ đơn lẻ. Ở các hệ thống Single-tasking, một khi tác vụ (process) được khởi động thì nó sẽ chạy hoàn tất trước khi tác vụ khác có thể được khởi động, ví dụ như hệ điều hành DOS.

- Để có thể thực thi một chương trình trên hệ điều hành single-tasking thường phải tốn nhiều thời gian chờ đợi cho những tác vụ có thời gian xử lý lâu như I/O, không phát huy tối đa khả năng CPU(vì phải chờ những tác vụ I/O).

- Để giải quyết vấn đề người ta đưa ra hệ điều hành multitasking. Như vậy multitasking được định nghĩa là việc thi hành đồng thời 2 hay nhiều tác vụ trên một CPU.

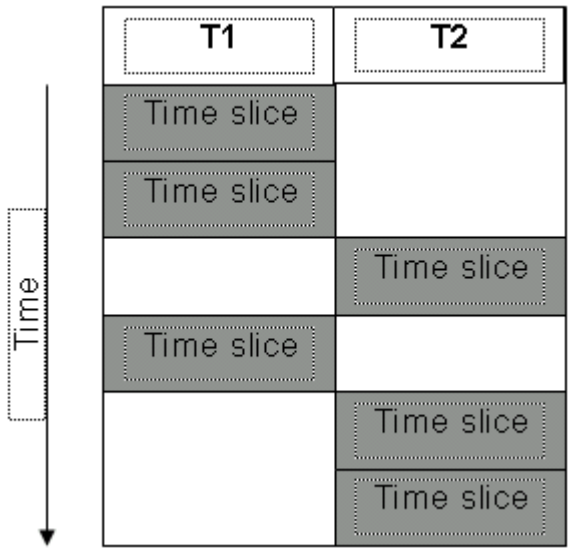
- Nhiều tác vụ khởi động để chạy trên một CPU. Hệ điều hành có trách nhiệm chuyển đổi các tác vụ thực thi trên CPU. Cách thức hệ điều hành điều khiển thi hành đồng thời nhiều tác vụ bằng cách gán cho CPU một tác vụ ở một thời điểm xác định. Hệ điều hành multitasking tạo nên ảo giác thi hành đồng thời bằng cách chia thành nhiều múi thời gian(time-slice).



- Khi có nhiều tác vụ thi hành cùng một lúc, mỗi tác vụ được CPU phục vụ trong một số lượng múi thời gian nhất định. Như vậy thực sự trong một thời điểm CPU chỉ phục vụ cho một tác vụ duy nhất, nhưng khoảng thời gian chuyển xử lý giữa các tiến trình rất nhỏ nên ta có cảm giác chúng được thi hành đồng thời.

Ví dụ:

T1,T2 là 2 tiến trình xử lý đồng thời.



- Tiến trình ưu tiên cho một tác vụ chạy sau một khoảng thời gian thực thi tác vụ khác được gọi là chuyển đổi ngữ cảnh(context switching).

- Các hệ điều hành multitasking có thể là ưu tiên (preemptive) hoặc không ưu tiên (nonpreemptive). Ở trường hợp hệ điều hành preemptive, ứng dụng không cần biết sự chuyển đổi giữa các tiến trình (sự chuyển đổi này được thi hành bởi hệ điều hành), sự khác nhau giữa hệ điều hành preemptive và nonpreemptive: ở hệ điều hành nonpreemptive là ứng dụng có nhiệm vụ từ bỏ CPU. Dạng multitasking cũng được tham khảo đến như là cooperative multitasking. Các hệ điều hành như Netware và windows 3.x là nonpreemptive multitasking, còn các hệ điều hành khác như MVS, VMS, UNIX,MAC, NT, và OS/2 là các hệ điều hành preeptive multitasking thật sự.

- Các tác vụ và tiến trình là các khái niệm cơ bản của bất kì hệ điều hành nào. Hệ điều hành phải có chức năng tạo huỷ các tiến trình.

b. Đa Luồng(multithreading)

❖ *Khái niệm luồng*

- Các chương trình truyền thống thực thi theo một kiểu tuần tự với một dòng (luồng, thread) điều khiển đơn độc, một sự nối tiếp các lệnh được thực thi bởi một tiến trình, một tiến trình nhiều hơn một luồng điều khiển được gọi là một tiến trình đa luồng (multithreaded processor). Một thread là một luồng điều khiển tuần tự đơn trong một chương trình, nó là sự nối tiếp các lệnh được thực thi trong một tiến trình.Mỗi thread có một điểm thực thi riêng lẻ. Các thread thường tham khảo đến như là các thread thực thi (thread execution), bởi vì các thread trong một tiến trình là kết hợp những lệnh nối tiếp nhau. Trong một chương trình đa luồng có thể có nhiều thread chạy đồng thời trong một không gian địa chỉ, mỗi thread có thể được xem như một processor ảo với bộ đếm chương trình(process counter),stack và tập thanh ghi riêng nó. Các thread là đơn vị cơ bản của sự thực thi sử dụng CPU.

- Mỗi thread trong một tiến trình chạy độc lập với các thread khác. Tất cả các thread trong một tiến trình chia sẻ một không gian địa chỉ chung và có quyền truy xuất ngang nhau đến tất cả các tài nguyên của tiến trình. Vì thread chia sẻ chung vùng không gian địa chỉ nên hành động của một thread có thể ảnh hưởng đến những thread khác trong một tiến trình.

- Khái niệm về thread và process là tương tự, một process có quyền sở hữu tài nguyên(thí dụ: memory, mở file,…), trong khi các thread là đơn vị có thể ra lệnh làm việc. Hầu hết các hệ điều hành multithread định thời các thread chạy trên một CPU. Nhiều thread không cần thiết chạy song song. Trên một đơn xử lý các thread được chia múi thời gian bởi hệ điều hành trong khi trên các máy có nhiều bộ xử lý các thread chạy song song trên nhiều bộ vi xử lý khác nhau.

- Thread hỗ trợ lập trình đồng thời và thường được dùng cho các tác vụ song song trong một trình ứng dụng. Các tiến trình quan trọng được thi hành song song trong một ứng dụng. Thread dễ tạo hơn là các tiến trình, và việc chuyển đổi ngữ cảnh cũng nhanh hơn các tiến trình. Vì việc duy trì ngữ cảnh của thread cũng nhẹ hơn nhiều so với process, cho nên thread còn được gọi là LightWeight Processes(LWP).

- Một việc rất quan trọng cần nhớ là một ứng dụng với rất nhiều công việc cần thực hiện mà nó chỉ có một thread sẽ chạy chậm hơn so với các máy có nhiều bộ xử lý cho đến khi ứng dụng được chia thành nhiều thread để thực thi. Một thread trong một tiến trình có thể chạy trên bất kỳ bộ xử lý nào và vì thế có khả năng khai thác tính song song vốn có của các máy có nhiều bộ xử lý.

❖ *Những thuận lợi khi dùng thread (Advantages of multithreading)*

- Tăng thông lượng và hiệu năng tốt hơn.

Các chương trình Multithreaded có thể thực thi trên môi trường Multiprocessor (MP). Trong môi trường MP, các thread như là tiến trình thực thi song song trên nhiều CPU như vậy thông lượng và sự thực thi tốt hơn. Thread có thể khai thác khả năng song song vốn có của các máy có nhiều bộ xử lý như vậy sẽ rút ngắn thời gian hoàn thành công việc.

Còn các máy đơn xử lý, các thread được cung cấp thông lượng tốt hơn bởi vì CPU có thể xử lý nhiều thread đồng thời khi có một vài thread bị block trên tác vụ I/O. Ví dụ một tiến trình có 2 thread thực thi trên máy đơn xử lý, khi một thread bị block trong khi gọi một hàm hệ thống (ví dụ file I/O), thì thread còn lại vẫn tiếp tục chạy.

- Thuật giải đơn giản

Nhiều tác vụ chương trình có thể mã hóa tốt hơn bởi các giải thuật trong các thành phần nhỏ và được phân cho các thread xử lý các thành phần này. Các thread cải tiến tốt hơn cho thiết kế chương trình

- Tính phản hồi cao(More responsive programs):

Bên cạnh những thread đơn, giao diện lập trình với người sử dụng được tính toán trong một thời gian dài, trong khi tiến hành sự tính toán, chương trình không thể tương tác với người sử dụng, bởi vì sử dụng các thread đơn để tính toán, thread chính hay GUI thread có thể thuận lợi cho người sử dụng.

- Tăng tính song song (Cheaper concurrency model):

Trong mô hình lập trình client – server, các chương trình server có thể xuất hiện như một thread xử lý đồng thời cho mỗi client.

❖ *Các khó khăn khi dùng thread*

- Added complexity(phức tạp).

- Difficult to debug and test(khó kiểm tra và debug).
- Data synchronization and race condition(sự đồng bộ dữ liệu và các điều kiện tranh đua).
- Potential for deadlock(khả năng bị deadlock).
- Non – thread – safe environment(môi trường thread không an toàn).

❖ *Mô hình tiểu trình(thread) trong JAVA*

- Có lẽ trong Java sức mạnh lớn nhất ngoài việc hướng đối tượng là khả năng multithreading. Điều đặc biệt là do support multithreading trong cả ngôn ngữ và các thư viện lớp nên việc sử dụng đặc tính này dễ dàng hơn rất nhiều.
- Khi dùng một ứng dụng single-thread, chỉ có thể tiến hành duy nhất một việc trong chương trình. Chương trình chiếm dụng tất cả tài nguyên của Java run-time system (dĩ nhiên điều này không có nghĩa là chương trình của ta là chương trình duy nhất chạy trên toàn bộ hệ thống. Tuy nhiên, đối với Java run-time system, chương trình của ta là thread duy nhất chạy trên máy ảo). Việc chạy các chương trình single-thread chỉ thích hợp cho những chương trình nhỏ, chỉ làm một nhiệm vụ đơn, còn trong thực tế yêu cầu của bài tóan conference là không thể: chương trình không thể đợi User nhập một câu Chat, chọn các User để gửi câu chat đi rồi mới listen message đến từ Server (ví dụ: tín hiệu logout, câu chat của user khác). Điều này cần được thực hiện đồng thời. Chính vì lý do đó mà việc sử dụng multithreading để hiện thực chương trình là bắt buộc.
- Hệ thống Java chạy dựa trên các thread và các lớp thư viện thiết kế với chức năng multithreading, Java sử dụng hiệu quả các tiểu trình này ngay trong môi trường không đồng bộ. Điều này làm giảm thiểu sự lãng phí CPU.
- Trong Java, có hai cách để tạo một lớp hiện thực như một thread:
  - ✓ Tạo lớp dẫn xuất từ lớp thread của java.
  - ✓ Cài đặt giao tiếp Runnable.

- Tạo một lớp là extends của lớp Thread.

```
class  className  extends Thread{
        public void run(){
    ... //Thread body of execution
        }
}
```

Khi gọi phương thức start(), phương thức run() tự động được gọi:

```
className myClass = new className();
myClass.start();
```

- Tạo một lớp implements Runnable interface

```
class  className  implements  Runnable{
        public void run(){
    ... // Thread body of execution
        }
}
```

Để chạy thread loại này, cần pass một instance của lớp cho một đối tượng Thread mới:

className myClass = new className();
new Thread(myClass).start();

❖ *Tính chất thread.*

- Java cho mỗi thread một độ ưu tiên trong tất cả các thread đang xử lý. Độ ưu tiên là một số nguyên cho biết thứ tự ưu tiên của nó với các thread khác. Độ ưu tiên của thread dùng để quyết định khi nào có thể chuyển sang thực hiện thread kế tiếp. Đây được gọi là chuyển đổi ngữ cảnh (context switch). Theo ngầm định, một thread thừa hưởng mức ưu tiên của thread cha. Ta có thể tăng hoặc giảm mức ưu tiên của bất kỳ thread nào bằng cách dùng phương t hức setPriority. Mức ưu tiên có thể được set trong khoảng giá trị từ MIN_PRIORITY (được định nghĩa là một trong lớp Thread) và MAX_PRIORITY (bằng 10). NORM_PRIORITY được định nghĩa là 5.

- Trong trường hợp 2 thread có cùng độ ưu tiên tranh giành CPU. Với hệ điều hành như windows 98 các thread có cùng độ ưu tiên được phân chia tự động. Với hệ điều hành khác như Solaris 2.x, các thread cùng cấp phải tự động nhường điều khiển cho thread khác. Nếu không làm điều này các thread khác sẽ không được chạy.

- Khi Thread-Scheduler có cơ hội nhận một thread mới, nó sẽ chọn thread có mức ưu tiên cao nhất hiện đang ở trạng thái runnable.

- Việc áp dụng threads rất hiệu quả khi thiết kế Client với đặc tính: luôn thực hiện "đồng thời" hai nhiệm vụ: vừa listen data do Server gửi cho, vừa tương tác với user. Ngoài ra Server cũng buộc phải hiện thực multithreading, mỗi thread quản lý một connection với một client.

❖ *Đồng bộ hóa các thread*

- Vì multithreading xử lý công việc không đồng bộ nên phải có cách đồng bộ hóa khi cần thiết. Ví dụ nếu bạn muốn hai thread liên kết và phân chia một cấu trúc dữ liệu phức tạp như danh sách liên kết, bạn cần vài cách chắc rằng chúng không đụng độ nhau. Bạn phải ngăn cản một thread đang ghi dữ liệu trong khi một thread khác đọc dữ liệu đó. Để thực điều này Java dùng kỹ thuật monitor. Monitor do C.A.R. Hoare đưa ra đầu tiên. Bạn có thể xem monitor là chiếc hộp nhỏ có thể giữ một thread. Một thread được nạp vào một monitor, tất cả các thread khác phải đợi cho đến khi thread đó thoát ra khỏi monitor.

- Hầu hết các hệ thống đa tiểu trình xem monitor như những đối tượng mà chương trình phải giành được. Trong Java không có lớp “Monitor”, mà có các đối tượng ẩn monitor được tự động tạo ra khi phương thức đồng bộ hóa được gọi. Khi một thread đã ở trong một phương thức đồng bộ, không có thread nào khác có thể gọi phương thức đồng bộ khác trong cùng một đối tượng. Điều này cho phép lập trình thread rất đơn giản và trong sáng.

❖ *Phương thức đồng bộ(synchronized)*

- Đồng bộ hoá rất dễ dàng trong Java vì các đối tượng đều có monitor đi kèm. Để truy xuất monitor của đối tượng chỉ cần gọi phương thức có thêm từ khoá synchronized. Trong khi một thread đang ở trong phương thức đồng

bộ hoá, tất cả các thread khác đang chờ cố gắng gọi phương thức này. Để thoát khỏi monitor và từ bỏ điều khiển của một đối tượng để nhận tiểu trình kế tiếp đang chờ, monitor dừng phương thức đồng bộ.

- Để hiểu sự cần thiết của việc đồng bộ hoá hãy bắt đầu với ví dụ đơn giản không cần đến việc đồng bộ này – những việc đồng bộ sẽ sử dụng.

- Như đã nói ở trên, khả năng multithread do Java support mang lại nhiều lợi điểm. Tuy nhiên, điều gì sẽ xảy ra nếu hai thread truy xuất và làm thay đổi cùng một đối tượng? Lý thuyết chung của vấn đề này vẫn là: phải đảm bảo trong quá trình một thread truy xuất và sửa đổi đối tượng dùng chung, nó không bị interrupted.

- Để giải quyết vấn đề này (crictical section), môn hệ điều hành có một số phương pháp: dùng semaphore, các giải thuật của Peterson, monitors, TESTANDSET. Tuy nhiên, Java xử lý vấn đề này bằng synchronize access tới các đối tượng dùng chung, đây là hình thức sử dụng monitors, tuy nhiên việc sử dụng trong Java lại rất dễ dàng, hầu như chỉ là vấn đề khai báo. Đơn giản là ta chỉ việc khai báo phương thức mà các thread gọi để truy xuất đối tượng dùng chung với từ khoá: synchronized

```
public synchronized void changeObject(...)
  {
   .....
  }
```

- Sau đây là cơ cấu làm việc của synchronization:

✓ Nếu một lớp có một hay nhiều phương thức synchronized, mỗi đối tượng của lớp nhận một hàng đợi, hàng đợi này giữ tất cả các thread đang đợi tới lượt thực thi một trong các phương thức synchronized.

✓ Có 2 khả năng để một thread xếp vào hàng: gọi phương thức synchronized trong khi thread khác đang sử dụng đối tượng dùng chung, hoặc chính thread đó gọi wait trong khi đang dùng đối tượng.

✓ Khi một lần gọi phương thức synchronozed trả về, hay một phương thức khác gọi wait, thread khác nhận quyền truy xuất tới đối tượng.

✓ Scheduler luôn chọn thread có mức ưu tiên cao nhất trong các thread đang trong hàng.

✓ Nếu một thread bị đặt vào hàng do gọi wait, nó cần được "unfrozen" bởi việc gọi notify trước khi nó được scheduled để thực thi tiếp.

- Các qui luật schedule khá phức tạp, nhưng sử dụng chúng lại khá đơn giản. Chỉ cần thực hiện ba qui tắc sau:

✓ Nếu hai hay nhiều thread sửa đổi một đối tượng, khai báo các phương thức thực hiện việc sửa đổi với từ khoá synchronized.

✓ Nếu một thread cần đợi sự thay đổi trạng thái của một đối tượng, nó nên đợi bên trong đối tượng, không phải bên ngoài, bằng cách vào phần thực thi của phương thức synchronized và gọi wait

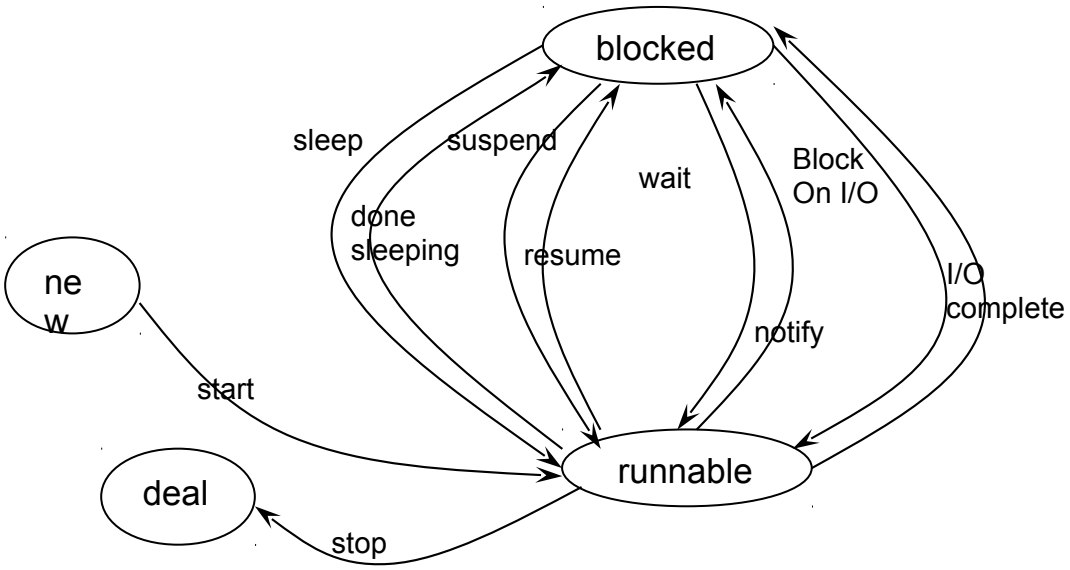
✓ Bất cứ khi nào một phương thức thay đổi trạng thái của một đối tượng, nó nên gọi notify. Điều này làm các thread đang đợi có cơ hội.

- Vấn đề này buộc phải xử lý trong đề án ở một số phương thức. Trong phần hiện thực Server, mỗi connection ứng với một client do một thread quản lý. Giả sử như có user_1 (client_1) thoát khỏi nhóm hiện hành của mình, khi đó thread_1 xử lý connection ứng với client này cần gọi hàm để xóa user khỏi nhóm. Nếu trong quá trình hàm xóa user khỏi nhóm đang thực thi mà thread bị interrupted, bởi một thread khác, thread_2, ứng với client_2 (user_2) cùng nhóm với user_1, muốn gửi một câu chat cho user_1. Sau đó thread_1 lại dành quyền điều khiển của thread_1, thực hiện xóa user_1 khỏi nhóm cũ. Thread_2 sau đó lại tiếp tục gửi câu chat cho user_1 lúc này không còn cùng nhóm với mình nữa. Vì thế buộc phải khai báo synchronized cho hàm xóa user khỏi nhóm.

❖ *Các trạng thái của thread*

Thread có ba trạng thái chính:

- ✓ Trạng thái sẳn sàng (ready)
- ✓ Trạng thái thực thi(running)
- ✓ Trạng thái block (waiting, sleeping, deal, blocked)



Các trạng thái của luồn

- Chu trình sống của một thread:Trước tiên thread được sinh ra (born), đưa vào trạng thái sẳn sàng (ready), tiếp tục vào trạng thái phục vụ(running), trong thời gian phục vụ nếu thread hoàn tất thì thread đó bị hủy bỏ, hoặc chờ sự kiện(có thể là I/O) nó được đưa vào các trạng thái tương ứng nếu sự kiện đang chờ xảy ra nó tiếp tục đưa vào trạng thái sẳn sàng(ready) để tiếp tục xử lý cho hoàn tất.

- Mỗi thread có một mức ưu tiên. Theo ngầm định, một thread thừa hưởng mức ưu tiên của thread cha. Ta có thể tăng hoặc giảm mức ưu tiên của bất kỳ thread nào bằng cách dùng phương t hức setPriority. Mức ưu tiên có thể được set trong khoảng giá trị từ MIN_PRIORITY (được định nghĩa là một trong lớp Thread) và MAX_PRIORITY (bằng 10). NORM_PRIORITY được định nghĩa là 5.

- Khi Thread-Scheduler có cơ hội nhận một thread mới, nó sẽ chọn thread có mức ưu tiên cao nhất hiện đang ở trạng thái runnable.

- Việc áp dụng threads rất hiệu quả khi thiết kế Client với đặc tính : luôn thực hiện "đồng thời" hai nhiệm vụ: vừa listen data do Server gửi cho, vừa tương tác với user. Ngoài ra Server cũng buộc phải hiện thực multithreading, mỗi thread quản lý một connection với một client.

9. Exceptions

- Java cũng support việc quản lý exception, một đặc trưng quan trọng tạo nên sức mạnh của chương trình. Bất cứ khi nào một lỗi run-time xảy ra trong một phương thức, nó có thể throw một exception, giúp đoạn code chứaphương thức đó khắc phục lỗi hay thoát có chủ ý, không làm down toàn bộ hệ thống. Điều này đặc biệt quan trọng khi chương trình của ta chạy cùng thời điểm với những ứng dụng quan trọng khác trong môi trường multitasking. Mặc dù hệ điều hành có thể đóng chương trình của ta mà không ảnh hưởng đến những ứng dụng khác, nhưng điều này không phải luôn đúng 100%.Việc quản lý exception được chú tâm đặc biệt khi hiện thực Server, vì Server là một chương trình cần phải chạy background, không thể bị down.

- Java code có thể dò lỗi và chỉ cho run-time system lỗi đó là gì. Thường thì một exception được throw sẽ làm cho thread gây ra lỗi đó kết thúc, và một thông báo lỗi được in ra. Nếu bạn muốn tự quản lý excetion, bạn có thể sử dụng phát biểu catch để bẫy exception.

- Đoạn code để bẫy một exception có thể xảy ra có dạng như sau:

```
Try{
... // some code which might throw an exception
}catch (exceptionType name){
  //Handle the exception
}
```

- Phần code trong khối try được thực thi và bất kỳ exception nào được tạo ra và match với phát biểu catch. Thực ra, bạn có thể dùng vài phát biểu catch khác nhau để bẫy những exception khác nhau. Bất kỳ phần code nào sau khi một exception được throw đều không được thực thi tiếp. Nếu phần code này buộc phải thực thi, bạn có thể sử dụng khối finally.

III. Java Server page(JSP)

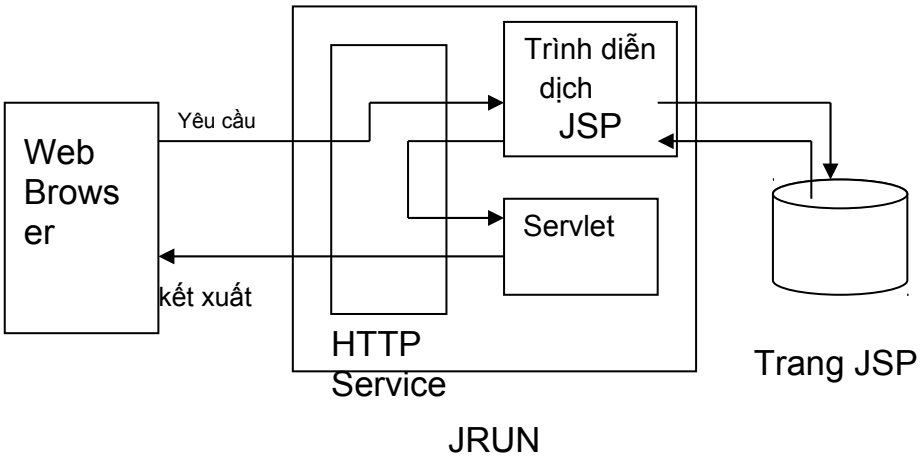
- JSP (Java Server Page) là cách kết hợp ngôn ngữ HTML truyền thống và ngôn ngữ Java phía máy chủ để sinh ra các trang web động phụ vụ các ứng dụng web uyển chuyển hơn các công nghệ CGI, Servlet.

❖ *Cơ chế hoạt động của JSP :*

- JSP đưa lệnh Java vào các mã (hay thẻ) HTML. Các trang JSP chứa các thẻ đặc biệt qui định gần giống như thẻ của ngôn ngữ HTML. Khi bạn yêu cầu một trang JSP, trình chủ sẽ đọc trang JSP từ đĩa cứng, bộ diễn dịch JSP (JSP Page Compiler) sẽ diễn dịch mã lệnh Java chứa trong trang JSP thành một servlet. Sau đó trình chủ Web Server sẽ triệu gọi servlet trả kết xuất thuần HTML về cho trình khách. Với trang JSP bạn hoàn toàn có thể lấy dữ liệu do

trình duyệt phía máy khách chuyển lên trình chủ xử lý sau đó gửi trả kết quả về cho trình khách.

- Cơ chế hoạt động của trang JSP được minh hoạ trong hình sau :



❖ *Các thẻ lệnh cơ bản của JSP*

- JSP cung cấp các thẻ lệnh giúp tạo trang web. Cũng tương tự như thẻ HTML, thẻ lệnh JSP bao gồm thẻ mở và thẻ đóng. Thực sự các thẻ JSP được xây dựng theo đặc tả và chuẩn XML (Extension Markup Language) nên có hơi khác với thẻ HTML vì chuẩn XML không xem chữ hoa và chữ thường giống nhau. Mỗi thẻ có các thuộc tính quy định cách dùng thẻ.

✓ Thẻ <jsp:scriptlet> hay <% %>

Thẻ này cho phép đặt các đoạn mã lệnh Java ở giữa cặp thẻ tương tự một chương trình java thông thường.

✓ Thẻ khai báo và thực hiện biểu thức <%! , <%=

Thẻ này dùng để khai báo một biến dùng cho toàn trang jsp. Biến khai báo phải đúng theo cú pháp của ngôn ngữ Java. Thẻ <%= được dùng để hiển thị một biểu thức.

✓ Thẻ nhúng mã nguồn <%@ include file %>

Với thẻ này có thể nhúng một trang. html vào trang jsp hiện hành. Thẻ này tương tự chỉ dẫn #include trong ngôn ngữ C. Cú pháp đầy đủ của thẻ này là:

<%@ include file = “URL or FilePath “ %>

✓ Thẻ chỉ dẫn biên dịch trang <%@ page %>

Thẻ này chỉ dẫn một số tính chất biên dịch áp dụng cho toàn trang jsp. Có thể sử dụng thẻ này để khai báo các thư viện import của java, chỉ định tuỳ chọn trang jsp có cần giữ trên cache bộ nhớ của trình chủ để tăng tốc hay không …

✓ Thẻ chuyển hướng <jsp:forward>

Thẻ này giúp chuyển hướng trang Web sang địa chỉ khác. Ví dụ, khi xử lý trang nhận dữ liệu đăng nhập (login page) bạn kiểm tra mật khẩu, nếu hợp lệ bạn chuyển người dùng đến trang tài nguyên cho

phép truy cập. Nếu không hợp lệ, bạn chuyển người dùng đến trang thông báo lỗi.

✓ Thẻ sử dụng thành phần Bean <jsp:useBean>

Bạn có thể tự tạo các lớp đối tượng Java và triệu gọi chúng từ bên trong trang jsp. Hướng theo công nghệ thành phần (component) Java gọi những đối tượng có thể gắn vào những ứng dụng là thành phần Bean.

✓ Thẻ đặt thuộc tính cho Bean <jsp:setProperty>

Thẻ này được sử dụng để triệu gọi một phương thức nào đó của Bean.

✓ Thẻ lấy thuộc tính của Bean <jsp:getProperty>

Ngược với thẻ <jsp:setProperty, thẻ <jsp:getProperty> dùng để lấy về nội dung của một thuộc tính.

❖ *Các đối tượng mặc định của trang JSP :*

- Trang diễn dịch JSP cho phép sử dụng một số đối tượng đã khai báo trước. Điều này giúp viết mã lệnh trong trang jsp nhanh hơn servlet.

- Đối tượng out: xuất phát từ lớp PrintWriter. Bạn có thể sử dụng đối tượng này để định dạng kết xuất gửi về máy khách. Ví dụ: <% out.println("Result"+7*3;%>

- Đối tượng request: xuất phát từ lớp HttpServletRequest. Đối tượng này giúp lấy về các tham số hay dữ liệu do trình khách chuyển lên.

- Đối tượng response: tương tự đối tượng out, đối tượng response dùng để đưa kết xuất trả về trình khách. Tuy nhiên, đối tượng out được dùng thường xuyên hơn. out hỗ trợ thêm luồng đệm để tăng tốc kiết xuất.

- Đối tượng session: xuất phát từ lớp HttpSesssion. Sử dụng đối tượng session để theo dõi kết nối và lưu vết một phiên làm việc giữa trình khách và trình chủ.

IV. Cơ sở dữ liệu trong Java

1. JDBC:

Java có hai hướng: là một ngôn ngữ lập trình và cũng là một hệ thống client/server trong đó chương trình tự động download và chạy trên máy cục bộ (thay vì máy server). Một trong những thu viện của API đó là Java Database Connectivity hay JDBC. Mục đích chính là kết nối chặt chẽ ngôn ngữ Java với cơ sở dữ liệu.

❖ *JDBC là gì ?*

- JDBC liên quan đến một vài thứ, tùy thuộc vào ngữ cảnh:

✓ Đó là một sự chỉ định rõ cho việc dùng tài nguyên data trong application và applet của Java.

✓ Đó là một API cho việc sử dụng JDBC cấp thấp.

✓ Đó là một API cho việc tạo driver JDBC cấp thấp, cái thực sự kết nối và chuyển đổi tài nguyên data.

✓ Đó là căn bản trên X/Open SQL Call Level Interface (CLI), định nghĩa làm thế nào sự tác động qua lại client/server là được thực thi cho hệ thống cơ sở dữ liệu.

- Java định nghĩa mọi hướng cho việc nhận dữ liệu của applicaton và applet driver của JDBC cấp thấp tiến hành việc chuyển cơ sở dữ liệu riêng biệt đến giao diện JDBC cấp cao hơn. Giao diện này được sử dụng bởi người phát triển và không cần lo lắng về cú pháp cơ sở dữ liệu đặc trưng khi tiến hành kết nối và query những cơ sở dữ liệu khác nhau. JDBC là một gói (package), giống như những gói khác của Java. Nhưng thông thường nó không phải là một phần của bộ phát triển phần mềm chuẩn, chẳng hạn như JDK. Các hướng hiện có của JDBC là những driver cần thiết cho việc kết nối những cơ sở dữ liệu mà không đòi hỏi bất cứ sự cài đặt nào trên client. Một driver JDBC có thể nạp xuống cùng một applet. JDBC chấp nhận những phần mềm cơ sở dữ liệu của các hãng sau :

♦Borland International, Inc.
♦Bulletproof.
♦Cyber SQL Corporation. DataRamp.
♦Dharma Systems, Inc.
♦Gupta Corporation.
♦IBM ‘ s Database 2 (DB2).
♦Imaginary (mSQL).
♦Imformix Software, Inc.
♦Intersoft.
♦Intersolv.
♦Object Design, Inc.
♦Open Horizon.
♦OpenLink Software.
♦Oracle Corporation.
♦Persistence Software.
♦Presence Information Design.
♦PRO-C, Inc.
♦Recital Corporation.
♦Rogne Wave Software, Inc.
♦SAS Institute, Inc.

❖ *Cấu trúc JDBC :*

Có những lý do để tách rời lập trình cấp thấp từ giao diện ứng dụng cấp cao. Lập trình cấp thấp là JDBC Driver. JDBC là rất uyển chuyển : nó có thể là tài nguyên dữ liệu cục bộ hay server cơ sở dữ liệu từ xa. Việc thực thi kết nối thực sự tài nguyên dữ liệu / cơ sở dữ liệu được dành cho bên trong JDBC driver.

Cấu trúc JDBC bao gồm những khái niệm sau :

Mục tiêu của JDBC là giao tiếp độc lập DBMS, một "cơ cấu truy xuất cơ sở dữ liệu SQL chung ", và một giao tiếp giống nhau cho tất cả các tài nguyên dữ liệu khác nhau. Người lập trình chỉ viết một giao diện cơ sở dữ liệu duy nhất : sử dụng JDBC, chương trình có thể truy xuất bất cứ tài nguyên dữ liệu nào.

Lớp DriverManager được sử dụng để mở một kết nối tới cơ sở dữ liệu qua JDBC driver, driver này phải đăng ký với DriverManger trước khi việc kết nối hình thành. Khi một kết nối được gắn vào, DriverManager lựa chọn từ một

danh sách các driver có thể tương thích với kiểu chính xác của cơ sở dữ liệu đã kết nối. Sau khi việc kết nối được hình thành, việc gọi query và lấy kết quả là được làm trực tiếp với JDBC
driver. JDBC driver phải thực thi những lớp để xử lý những hàm cho cơ sở dữ liệu riêng biệt, nhưng đặc điểm kỹ thuật của JDBC đảm bảo rằng driver sẽ được tiến hành như dự kiến. Điều cốt yếu là người phát triển có JDBC driver cho cơ sở dữ liệu không cần thiết phải lo lắng về việc phải thay đổi đoạn mã cho chương trình Java nếu một kiểu cơ sở dữ liệu khác được sử dụng (giả sử rằng JDBC driver cho những cơ sở dữ liệu khác là có sẵn). Điều này đặc biệt hữu dụng cho các cơ sở dữ liệu phân bố.
JDBC sử dụng cú pháp URL cho việc chỉ định một cơ sở dữ liệu. Ví dụ một kết nối tới một cơ sở dữ liệu mSQL sẽ có danh như sau :
jdbc : msql://mydatabase.server.com:1234/testdb

Câu lệnh này chỉ định "phương tiện" được sử dụng (jdbc), kiểu cơ sở dữ liệu (msql), tên server, cổng (1234), và cơ sở dữ liệu được kết nối tới (testdb)

Kiểu dữ liệu trong SQL được ánh xạ vào kiểu nội bộ Java bất cứ khi nào có thể. Khi một kiểu nội bộ là không miêu tả trong Java, một lớp là có sẵn trong việc nhận dữ liệu kiểu đó. Ví dụ, kiểu Date trong JDBC. Một người phát triển có thể gán một field ngày trong cơ sở dữ liệu với lớp ngày JDBC, sau đó người phát triển có thể sử dụng phương thức trong lớp Date để hiển thị hay tiến hành các thao tác. JDBC cũng bao gồm các đối tượng nhị phân, do đó chúng ta có thể nhận và lưu trữ ảnh, nhạc, tài liệu, hay những dữ liệu nhị phân khác trong cơ sở dữ liệu với JDBC.

2. *ODBC và JDBC :*

ODBC và JDBC chia sẻ nguồn gốc chung : cả hai là nền tảng trên X/OPEN gọi cấp giao tiếp cho SQL. Mặc dù những JDBC driver nổi bật cho nhiều cơ sở dữ liệu, chúng ta có thể viết chương trình Java sử dụng ODBC driver hiện có. Trên thực tế là Javasoft và Intersolv có viết một Java driver - cần nối JDBC - ODBC - cho phép người phát triển sử dụng ODBC driver hiện có trong chương trình Java. Hình bên trên minh hoạ ví trí của JDBC - ODBC Brigde trong kiến trúc toàn thể của JDBC. Tuy nhiên JDBC - ODBC Brigde đói hỏi sự cài đặt trên client, hay ở nơi mà chương trình Java thực sự chạy, bởi vì Bridge phải gọi phương thức nội bộ để chuyển từ ODBC sang JDBC. Chỉ có Java driver 100% mới có thể download thông qua mạng với một Java applet, không cần bất cứ một cài đặt lại nào. Nhiệm vụ của ODBC driver trong kiểu tương tự như JDBC driver. Trên thực tế JDBC - ODBC Brigde thật sự là một JDBC driver chuyển đến và từ ODBC ở cấp thấp. Khi JDBC driver cho cơ sở dữ liệu sẵn có, chúng ta có thể dễ dàng bật từ ODBC driver đến JDBC driver với một vài thay đổi, nếu có, sự thay đổi đoạn mã của chương trình Java.

❖ *Sử dụng JDBC driver :*

Muốn sử dụng JDBC driver, đầu tiên chúng ta phải có JDBC driver (vì chúng không đi kèm với những gói có trong Java API). Tiếp theo chúng ta muốn sử dụng ODBC, đừng quên rằng chúng ta sẽ cần ODBC driver. Nếu chúng ta không có server cơ sở dữ liệu, nhưng muốn sử dụng JDBC, chúng ta có thể sử dụng gói ODBC driver với Microsoft Access. Sử dụng JDBC -ODBC

Bridge, chúng ta có thể viết những Java application giao tiếp với một cơ sở dữ liệu Access. Không may, applet bắt buộc sự giới hạn bảo mật nên không cho phép truy xuất đĩa cục bộ, vì vậy ODBC driver có
thể không làm việc trong phạm vi applet (trên một trình duyệt Web). Tuy nhiên có thể bây giờ, với sự thay đổi công nghệ có thể có khả năng sử dụng JDBC - ODBC Bridge. Sử dụng ODBC driver trong chương trình Java đòi hỏi sự cài đặt lại ODBC driver và JDBC - ODBC Bridge trên máy client. Ngược lại, JDBC driver là lớp Java có thể download động qua mạng cùng với file chứa applet được gọi.

Đăng ký và gọi JDBC driver :

+ Sau khi cài đặt những lớp JDBC, muốn sử dụng JDBC driver chúng ta phải import.java.sql.* vào chương trình Java của mình. Những lớp JDBC căn bản có chứa những thành phần cần thiết cho JDBC driver, và chúng phục vụ như người đứng giữa chúng ta và mã cấp thấp trong JDBC driver. JDBC API cung cấp chúng ta với một giao diện dễ sử dụng cho việc giao tiếp với tài nguyên dữ liệu, độc lập với driver mà chúng ta sử dụng.

❖ *Sử dụng JDBC để truy xuất cơ sở dữ liệu :*

Trước khi có thể truy xuất tới cơ sở dữ liệu ODBC dưới nền Window95/98/Me hay WinNT/Win2000 Server, ta phải đăng ký nó với một bảng điều khiển driver ODBC. Dưới nền Window95/98/Me, đó là một icon ODBC trong chương trình Control Panel. Còn trong WinNT, nó được tìm thấy trong Start menu.

Nhấp đôi chuột vào icon *ODBC*, sau đó chọn mục "Add" như hình bên dưới.

Sau đó chọn một driver *cơ sở dữ liệu* (ở đây ta chọn Microsoft dBase Driver) và nhấp vào "OK". Đánh tên *cơ sở dữ liệu* vào Data Source Name và Description, rồi chọn mục "Select" để cập nhật và chọn nó. Sau khi cập nhật *cơ sở dữ liệu*, màn hình sẽ hiển thị như ở hình bên dưới.

3. Kết nối tới* Cơ sở dữ liệu*:

Tất cả các đối tượng *cơ sở dữ liệu* và phương thức được đặt trong gói java.sql, và do đó ta phải import gói java.sql.* vào chương trình đang sử dụng *JDBC*. Để kết nối tới một *cơ sở dữ liệu ODBC*, trước tiên ta phải load cầu nối driver *JDBC_ODBC*:

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Phát biểu này sẽ load driver và tạo ra một đối tượng của lớp đó. Sau đó, để kết nối tới tới một phần của *cơ sở dữ liệu*, ta phải tạo ra một đối tượng của lớp Connection, và quy cho *cơ sở dữ liệu* sử dụng cú pháp URL:

String url="jdbc:odbc:Subname");

Connection conn=create.getConnection(url);

Subname chính là Data Source name mà ta đăng ký trong Control Panel.Cú pháp URL có thể biến đổi hoàn toàn cho các kiểu khác của *cơ sở dữ liệu*.

jdbc.Subprotocol.Subname

Những chữ đầu tiên minh hoạ cho protocol kết nối và luôn luôn là jdbc. Subprotocol ở đây được ghi rõ là odbc. Nó định nghĩa cơ chế kết nối cho một lớp *cơ sở dữ liệu*. Nếu ta kết nối tới một Server *cơ sở dữ liệu* trên một máy

khác, ta có thể ghi rõ tên máy, Subprotocol đó cũng như UserName, password như một phần của chuỗi Connection.

4. Truy suất Cơ sở dữ liệu trong java*:*

Một khi đã kết nối được tới *cơ sở dữ liệu*, điều ta đòi hỏi là các thông tin trên những tên bảng, tên cột, nội dung của các cột, và ta có thể chạy SQL mà không cần phải truy vấn đến *cơ sở dữ liệu*, hoặc thêm vào, hoặc sửa chữa lại nội dung của nó. Các đối tượng mà ta có thể sử dụng để thu được các thông tin từ *cơ sở dữ liệu* là:

♦DatabaseMetaData: thông tin đầy đủ về *cơ sở dữ liệu*: tên bảng, tên các chỉ mục của bảng, tên sản phẩm *cơ sở dữ liệu*, version và các hoạt động được cung cấp bởi *cơ sở dữ liệu*.

♦ ResultSet: thông tin về một bảng hay kết quả của một truy vấn. Ta có thể truy xuất tới hàng dữ liệu bởi một hàng, nhưng có thể truy xuất tới các cột bằng nhiều cách khác.

♦ResultSetMetaData: thông tin về một tên cột, các kiểu của cột trong đối tượng ResultSet.

Trong khi mỗi đối tượng này có một số lượng lớn các phương thức hướng dẫn ta lấy được các thông tin chi tiết chính về các phần tử của *cơ sở dữ liệu*. Có một vài phương thức chính trong mỗi đối tượng này cho ta những thông tin đầy ý nghĩa nhất về *cơ sở dữ liệu* của ta.

CHƯƠNG 5

PHÂN TÍCH, XÂY DỰNG VÀ CÀI ĐẶT CHƯƠNG TRÌNH MAIL SERVER TRÊN MÁY SERVER

Chương trình được thiết kế bằng ngôn ngữ Java của hãng Sun, hỗ trợ các giao thức SMTP, POP3, IMAP4, đa miền(Domain),… Lưu trữ thông tin người dùng được tổ chức theo dạng chuẩn mà các mail server hiện nay được dùng là dưới dạng cây thư mục. Còn thông tin về người sử dụng được lưu trữ trong cơ sở dữ liệu bằng Access thông qua cầu nối ODBC. Chương trình chạy trên mọi môi trường và dùng bất kỳ một trình mail client nào cũng có thể truy xuất đọc và gởi mail được cả. Chương trình được viết dưới dạng đa luồng nên không ảnh hưởng đến tốc độc truy xuất, khả năng xử lý cho từng người dùng khi kết nối vào cùng một lúc. Chương trình rất thân thiện với người dùng dễ hiểu và hoàn toàn đáp ứng một dịch vụ mail Server hỗ trợ đa miền.

I.Nhận xét về các giao thức mail và các gói trong chương trình.

1.Nhận xét về giao thức.

- Giao thức gởi thư trên Internet hiện nay chỉ duy nhất có một đó là giao thức SMPT được trình bày ở phần trên. Nhìn chung giao thức này cũng không có gì đáng nói, những lệnh mà client gởi đến phải tuân theo đúng qui luật của giao thức. Nghĩ là lệnh phải có lệnh nào được quyền thực hiện trước lệnh nào thông thương là lệnh HELLO (HELO) tiếp theo là MAIL, RCPT, DATA còn những lệnh khác được sử dụng tuỳ theo từng trường hợp mà người sử dụng cần đến để có được một kết quả tốt. Trong chương trình cơ bản em đã mô phỏng cơ bản là đầy đủ các lệnh trong giao thức SMTP, giao thức này được tìm thấy trong gói SMTP của chương trình nguồn, gói này đảm nhận việc nhận mail và lưu trữ mail đúng địa chỉ nếu nó thuộc quyền quản lý của mail server còn không nó sẽ lưu vào hộp thư outbox và gọi lại chương trình SPTP reply hay SMTP Forwad ra thực hiện tiếp tuỳ theo cấu hình trong chương trình. Nhìn chung giao thức này được ứng dụng trong chương trình rất tốt, đảm bảo độ chính xác an toàn và thư gởi đến không bị thất lạc.

- Giao thức nhận mail hiện nay có hai giao thức là POP và IMAP, mỗi giao thức có những ưu và khuyết điểm riêng cụ thể như sau.

*** Sự khác biệt giữa hai giao thức**

- Nói một cách đơn giản nhất, IMAP đặt sự kiểm soát e-mail lên server trong khi nhiệm vụ duy nhất của POP là "ném" toàn bộ thông điệp e-mail về chỗ trình client yêu cầu, và xong là "phủi tay". Theo nhận định của *Terry Gray* Giám đốc Networks & Distributed Computing ở University of Washington, và là người dùng IMAP 5 năm nay phát biểu: "Nhu cầu truy cập đến một nơi lưu trữ thư duy nhất từ nhiều máy khác nhau vào những thời điểm khác nhau là lý do chủ yếu cho sự bùng nổ mối quan tâm đến IMAP, và là nguyên nhân tại sao mọi nhà cung cấp hệ thống thư điện tử có tầm cỡ hiện nay đều đang hoặc sẽ hỗ trợ nó". Qua đó, cho ta thấy được giao thức IMAP vẫn là giao thức đọc mail tốt và an toàn nhất. Cụ thể, IMAP cung cấp truy cập e-mail theo ba chế độ khác nhau: offline (ngoại tuyến), online (trực tuyến), và disconnected (ngắt kết nối).

+Truy cập ở chế độ offline chính là POP, trong đó các thông điệp được truyền đến máy client, xoá khỏi server, và mối liên kết bị ngắt. Sau đó người dùng đọc, trả lời, làm các việc khác ở chế độ ngoại tuyến, và nếu muốn gửi thư mới đi họ phải kết nối lại.

+ Truy cập online, như tên gọi của nó, là chế độ truy cập mà người dùng đọc và làm việc với thông điệp e-mail trong khi họ đang kết nối với server (kết nối mở). Các thông điệp này vẫn nằm ở server cho đến khi nào người dùng quyết định xoá nó đi. Chúng đều được gắn nhãn hiệu cho biết loại để "đọc" hay "trả lời".

+ Trong chế độ disconnected người dùng lưu tạm thông điệp ở client, làm việc với chúng, sau đó cập nhật trở lại vào server ở lần kết nối sau. Chế độ này hữu ích cho những ai dùng laptop hay vào mạng bằng liên kết quay số điện thoại, đồng thời không muốn bỏ phí những lợi điểm của kho chứa thư ở server.

- Hơn nữa, với các thông điệp e-mail được lưu giữ ở server, tất cả các các thao tác trên thông điệp đều có thể thực hiện được, chẳng hạn như có thể chỉ đọc phần header của thông điệp (xem thêm ở danh sách tính năng dưới đây).

Ngoài khả năng thao tác trên thông điệp, dùng IMAP còn có những lợi ích khác nữa. Ví dụ, khi người dùng làm việc trên hai hay nhiều máy PC, chẳng hạn một máy để bàn, một máy di động, họ không cần phải lo lắng về việc thư từ nằm vương vãi trên các máy client khác nhau. Cũng vậy, đối với những cơ quan có người dùng di động và cả những máy PC để cố định, chẳng hạn như các trường học hay cơ quan y tế, IMAP là một giải pháp lý tưởng ố theo ý kiến của một chuyên gia phụ trách công nghệ.

- Một lợi điểm khác của IMAP là e-mail có thể được tự động sao dự phòng trong server của xí nghiệp và tại máy cá nhân. Những thông điệp quan trọng sẽ được người dùng chép về máy client, và bản sao vẫn để lưu tại máy server. Tuy nhiên, phần lớn người dùng có khuynh hướng để lại các thông điệp ở server. Làm như vậy sẽ giúp cơ quan có thể có kế hoạch tổ chức việc sao dự phòng (backup) cho toàn bộ e-mail của cơ quan một cách dễ dàng. Tất nhiên việc quyết định tổ chức sao dự phòng tập trung phụ thuộc chủ yếu vào mức độ quan trọng của nội dung e-mail, nhưng nếu e-mail nằm rải rác trên các máy client thì khi cần thực hiện sao dự phòng bạn sẽ gặp rất nhiều khó khăn.

- IMAP cũng làm giảm nhu cầu mà e-mail đòi hỏi trên mạng. Người dùng trước đây phải tải xuống toàn bộ e-mail họ có với POP, dù e-mail đó có chứa file gắn kèm lớn nhiều megabyte, thì giờ đây họ có thể tải xuống một cách có chọn lọc toàn bộ hay một phần nào đó của bức thư. Kết quả, theo ý kiến các nhà cung cấp, thì phần lớn người dùng ít tải thư về hơn. Kỹ thuật e-mail trên server còn có nghĩa là chi phí thiết lập server sẽ tăng lên khá nhiều so với POP.

*** Những tính năng xuất sắc nhất của IMAP**

- Thật ra thì nhiều tính năng trong danh sách sau đây có thể có trong các trình client e-mail riêng. Nhưng hãy nhớ rằng Internet Messaging Access Protocol (IMAP) cung cấp một thứ mà các chương trình kia không có: độc lập với nhà cung cấp, và đang được chuẩn hoá.

+ Người dùng có thể chỉ cần xem phần header của thông điệp để xác định thông điệp nào cần đọc.

+ IMAP có thể chèn các thông điệp vào folder ở xa.

+ Nó cho phép tạo ra các nhãn hiệu chuẩn hay được định nghĩa bởi người dùng cho thông điệp. Ví dụ, nhãn hiệu có thể dùng để định danh các nhóm làm việc, các dự án, v.v...

+ Do các thông điệp IMAP có khuynh hướng được giữ lại tại server chứ không phải ở các máy client riêng lẻ, có thể cập nhật được chúng. IMAP hỗ trợ cập nhật đồng thời trong các folder dùng chung và thông báo cho người dùng về việc cập nhật.

+ Người dùng có thể có nhiều folder trong một hộp thư đến (inbox), và có thể thiết lập chúng theo nhiều cách, như tạo cây thư mục.

+ Người dùng có thể lựa chọn để đọc các phần của thông điệp MIME, như truy cập phần thân của thông điệp và bỏ qua phần gắn kèm.

+ Người dùng có thể tìm kiếm các thông điệp trên server.

2.Các gói trong chương trình.

Chương trình có tổng cộng là 10 gói và hai gói đi kèm của hãng Sun là mail.jar và activation.jar, ngoài ra có file cấu hình server là config.ini.

- Config.ini: đây là file lưu lại các thao tác thay đổi trên chương trình chính để lần sau khi chạy chương trình, chương trình sẽ lấy lại cấu hình đó nếu file này bị xoá chương trình sẽ tạo ra file khác và lấy lại cấu hình mặt định được cài đặt trong chương trình. File này rất quan trọng và không thể thiếu được.

- Mail.jar : đây là gói của hãng sun đi kèm theo trong trình dịch java, ứng dụng của gói này là dùng để hỗ trợ cho việc đọc mail và gởi mail dựa vào gói này người lập trình viên có thể viết một chương trình client như đọc mail và gởi mail một cách dễ dàng thông qua các lớp sẵn có trong gói tin.

- activation.jar : gói này chủ yếu là sử lý các biến cố liên quan tới mail như khi đọc mail thì gói này làm nhiệm vụ phân tích mail đó có file đi kèm không hay mail được định dạng theo kiểu nào, cò khi gởi mail thì cần gọi các thủ tục trong gói tin này ra để định dạng mail theo một dạng chuẩn rồi tiến hành gởi đi. Ngoài ra còn nhiều tính năng khác tuỳ theo mục đích sử dụng trong chương trình.

- gói Servermail : đây là gói điều khiển chính của chương trình và tương tác với người sử dụng gói này có tổng cộng là 166 lớp và 2 gói nhỏ là Domain(là những lớp điều khiển tên miền như xoá, thêm hay cập nhật) và Newmail(các lớp trong gói này sử lý việc gởi mail). 166 lớp được dịch ra từ 48 lớp chương trình nguồn, trong đó lớp chính để chạy là server.class.

- Gói ClassStore : đây là gói tiễn ích dùng chung cho các gói khác.

- Gói Imap4 : gói này cấu tạo nên giao thức IMAP4, xử lý tất cả các lệnh liên quan đến giao thức có tốc độ truy cập tương đối nhanh, người dùng không chờ đợi lâu. Gói này có tổng cộng là 5 lớp trong đó lớp chính là ImapServer.class. khi có một người sử dụng kết nối vào thì lớp ImapThread.class được tạo ra phục vụ riêng cho cho người dùng đó đến khi đóng kết nối thì lớp này sẽ được giải phóng và trả lại vùng nhớ đã lưu trữ trước đó.

- Gói POP3: tương tự như gói Imap gói này cấu hình nên giao thức Pop3, xử lý các lệnh liên quan đến pop3, có tổng cộng 4 lớp, lớp chính là POPServer.class. khi có kết nối từ người dùng lớp POPConnection.class được khởi tạo riêng cho người dùng đó và được giải phóng khi kết nối đó không còn hiệu lực.

- Gói SMTP: cấu tạo nên giao thức SMTP, có tổng cộng 4 lớp trong đó lớp chính là SMTPServer.class, lớp SMTPConnection.class được tạo ra khi có kết nối từ người dùng và được giải phóng sau khi kết nối.

*** Chức năng Replay hay Forward.**

- Có một vấn đề cần nói đến là khi trình mail Server nhận được một lá thư mà người nhận thư không thuộc quyền quản lý của trình mail server đang chạy, làm cách nào để đảm bảo thư đến được người dùng?.Cụ thể là với một địa chỉ mail như name@yahoo.com hay name@hotmail.com làm cách nào biết được địa chỉ mail server thực sự?. Chúng ta dựa vào tên domain mail. Mỗi mail Server sẽ đăng ký với DNS server nhưng tên miền mà nó quản lý theo dạng MX record(Mail Exchange Record). Ngoài ra ta có thể dùng một mail server trung gian để chuyển mail đến đích.

+ ServerReply: là trình server mail đang chạy sẽ chuyển nội dung lá thư mà địa chỉ mail đến mà không thuộc quyền quản lý thông qua một mail Server khác có hỗ trợ dịch vụ này, thông thường thì rất ít trình Mail Server hỗ trợ dịch vụ này. Vì việc này sẽ làm cho trình chủ bị quá tải và tính bảo mật sẽ không

được an toàn. Trên mạng internet các mail server của vnn.vn có hổ trợ dịch vụ server mail trung gian này.

+ ServerForward: đây là một phương pháp tối ưu và hiệu quả nhất. Cụ thể là khi nhận được một lá thư mà địa chỉ đến không thuộc quyền quản lý, trình mail server đang chạy sẽ phân tích xem Domain mail này thuộc quyền quản lý của Server mail nào và địa chỉ máy đang chạy tên gì thông qua máy server DNS cổng kết nối là 53 (DNS Server lưu trữ các địa chỉ máy server đang chạy trên mạng Internet, ở việt nam có 2 Server DNS tên là hcm-server1.vnd.net và dng-server2.vnd.net). Sau đó tiến hành gởi mail đến người nhận thông qua server mà nó đang quản lý.

Ví dụ: ServerForward có tính năng tương tự nhu Nslookup.exe của dòng họ windows server. Sau đây là một ví dụ phân giải tên domain yahoo.com thành địa chỉ mail server thực thụ bằng chương trình nslookup.exe trong windows Server hay MXLookup.class trong chương trình là như nhau.

```
C:\winnt> nslookup
  Default Server: hcm-server1.vnd.net
  Address: 203.162.4.1
 >set type=MX
 >yahoo.com
  yahoo.com   MX preference=1, mail exchanger = mx1.mail.yahoo.com
  yahoo.com   MX preference=1, mail exchanger = mx2.mail.yahoo.com
  yahoo.com   MX preference=5, mail exchanger = mx4.mail.yahoo.com
```

từ ví dụ trên ta thấy domain mail yahoo.com có tới 3 server mail quản lý đó là mx1.mail.yahoo.com, mx2.mail.yahoo.com và mx4.mail.yahoo.com. Ta sẽ chon ra một mail Server trong 3 mail server trên và gởi thư đến mail server này khi người nhận mail có domain mail là yahoo.com.

Trong chương trình các gói tin sau sẽ làm những nhiệm vụ trên.

+ NSLookup : đây là gói đảm nhận nhiệm vụ chuyển tiếp mail. Nghĩa là khi có một lá thư được nhận mà địa chỉ người nhận không thuộc quyền quản lý của Server, chương trình chính sẽ gọi lớp SMTPForwardServer.class ra thực hiện. Lúc này Chương trình chạy trên server đóng vai trò là máy trạm(Client) chuyển tiếp thư đến người nhận thông qua một server mail khác. Trong chương trình Server chuyển tiếp mặt định là Smtp.hcm.vnn.vn đây là một mail Server của Vnn.vn đặt tại thành phố Hồ Chí Minh, thông qua server này mail được chuyển đến đúng người nhận.

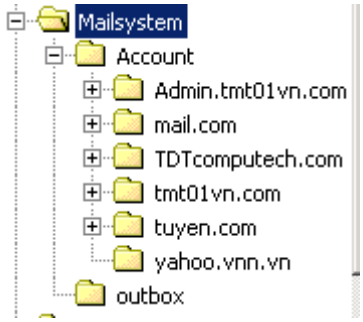
+ MXLookup: đây là gói đảm nhận việc phân tích và giả mã domain mail mà không thuộc quyền quản lý của server thông qua DNS Server(DNS mặc định trong chương trình là hcm-server1.vnd.net) và tiến hành gởi thư đến Server mà nó quản lý. Có tổng cộng là 8 lớp trong đó lớp chính là SMTPRelayServer.class làm nhiệm vụ gởi mail và lớp NSLookup.class lấy về tên máy server đang quản lý domain mail cần gởi đến.

- Gói StoreUser : đây là gói tin dùng để lưu trữ và tương tác với ổ đĩa trên máy như lưu trữ mail hay thông tin người dùng trong cơ sở dữ liệu,…..
- Gói Systray : xử lý đưa biểu tượng xuống Systray nếu hệ điều hành đang chạy là Windons.
- Gói Tree : gói tiễn ích tạo cây thư mục.

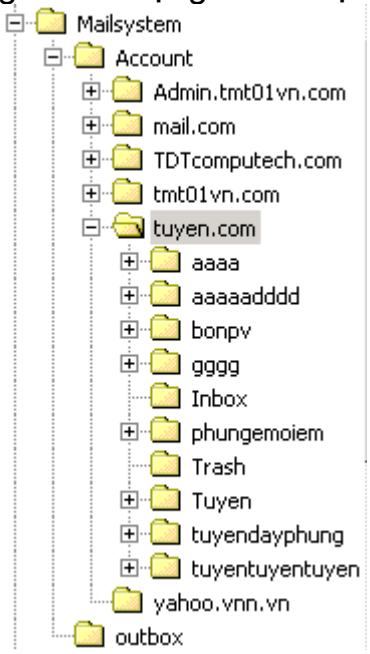
II. Mô hình cây thư mục lưu trữ mail trên máy:

- Thư được lư trữ dưới dạng cây thư mục, thư mục gốc được mặt định là c:\MailSystem, bên trong có 2 thư mục con, một là Account và outbox.

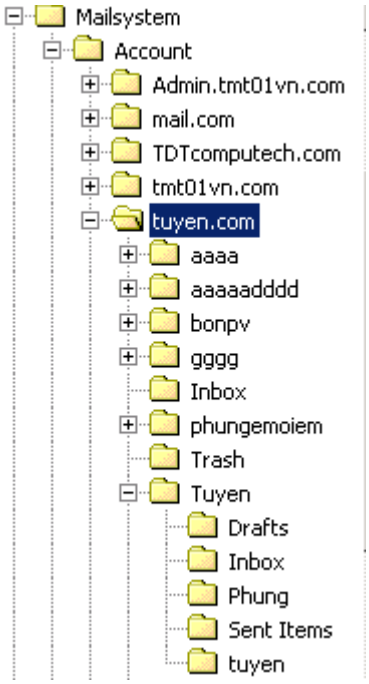
+ Account: là thư mục chính nó lưu trữ các domain và Account người sử dụng được mô tả ở bên dưới



bên trong thư mục Accout có 6 thư mục, tên từng thư mục tương ứng với tên từng Doain mà server đang quản lý. Bên trong các thư mục Domain này có rất nhiều các thư mục con và tên thư mục con trong từng Doamin tương ứng với tên Account mail của một người sử dụng nào thuộc Domain nay.



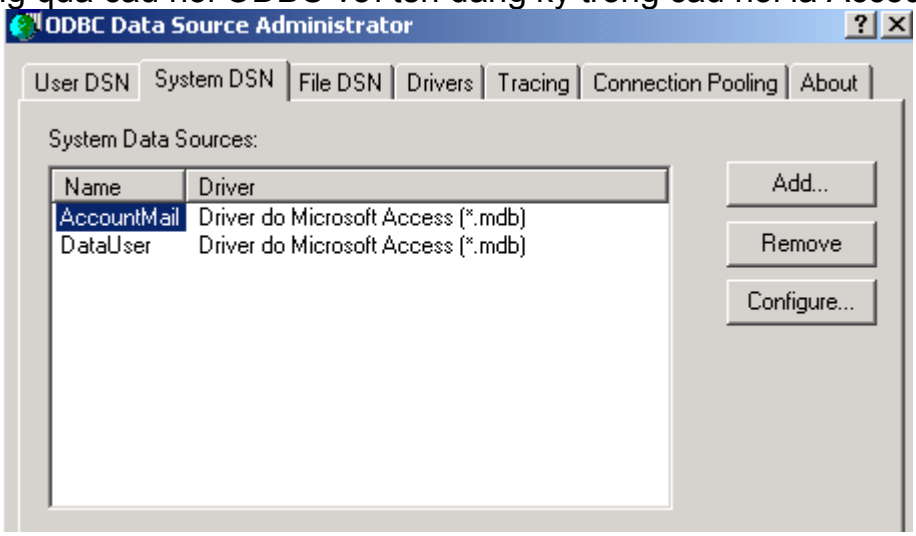
Ví dụ thư mục tuyen thuộc Domain tuyen.com như vậy địa chỉ mail có tên là tuyen@tuyen.com và tên của Account là tuyen. Vào bên trong từng Account này sẽ có ít nhất 4 thư mục chủ yếu của giao thức Imap là Ibox, Trash, Drafts, Sent Items, và các thư mục khác do người sử dụng tạo ra, tên những thư mục này tương ứng với tên thùng thư mà chúng ta thấy trong duyệt mail từ Client và bên trong mỗi thư mục chứa các tập tin thư, thông thường thư đến sẽ được lưu trữ trong thư mục Inbox.



- Thư mục Outbox: đây là thư mục lưu trữ nội dung của những lá thư đến không thuộc quyền quản lý được lưu tạm vào đây chờ gọi chương trình ServerReply hay ServerForward ra xử lý.

III.Lưu trữ thông tin người dùng :

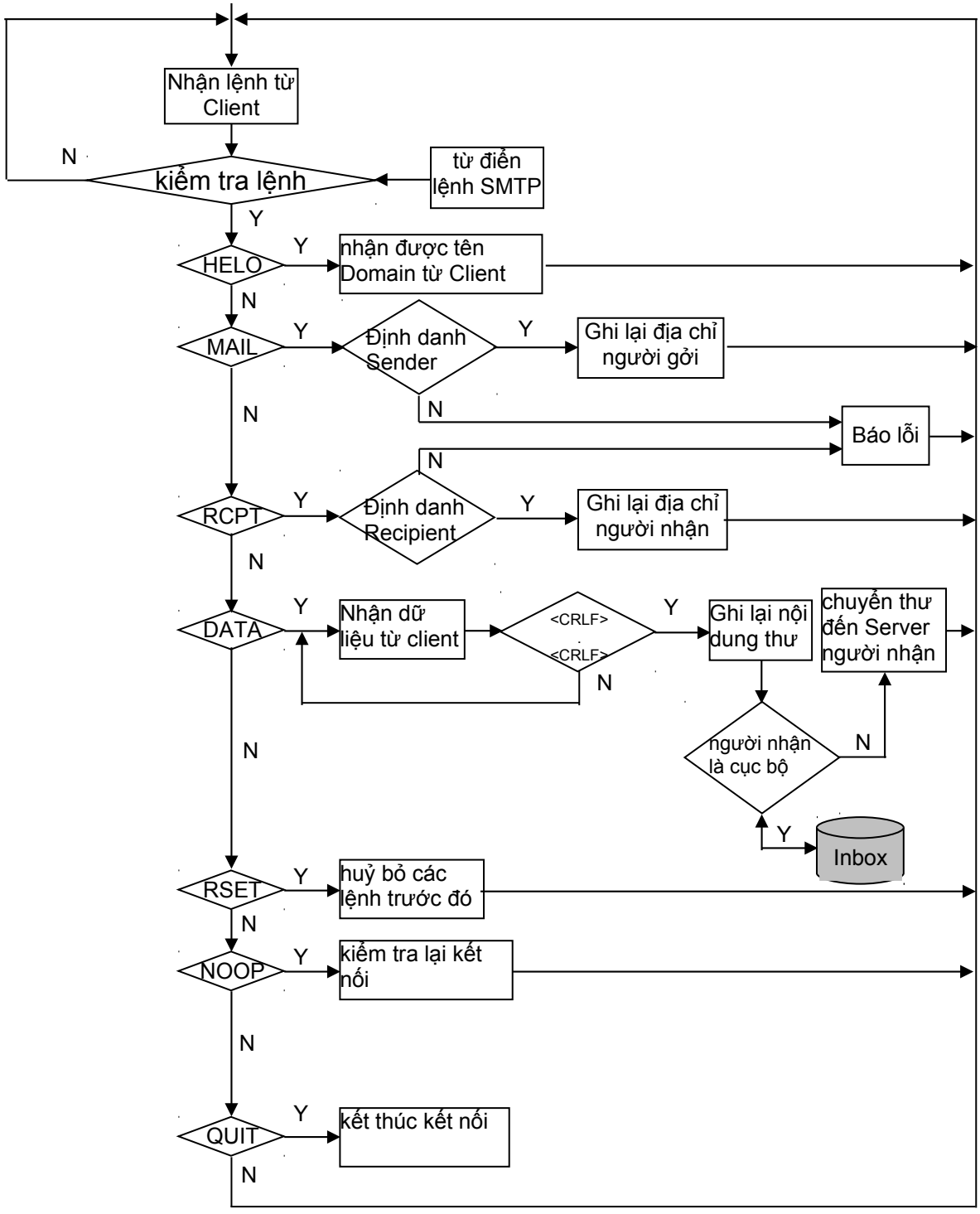
- Để cho tiện trong quá trình thao tác cũng như cải thiện tốc độ xử lý trong chương trình em dùng cơ sở dữ liệu là Access để lưu trữ thông tin người dùng thông qua cầu nối ODBC với tên đăng ký trong cầu nối là AccountMail



Cơ sở dữ liệu này lưu trữ cơ bản là những thông tin mà người dùng đăng ký ban đầu hay là lần sửa đổi gần đây nhất. Bao gồm Họ, tên, Tên Account, Ngày sinh, Address book,…… Mục đích chính là cho điện trong việc quản lý người dùng cũng như việc cập nhật, thêm hay loại bỏ người sử dụng. Ngoài ra còn một số lưu trữ khác nhưng mục đích chính của em là dùng để lưu trữ thông tin nên các table hay các trường trong cơ sở dữ liệu không có ràng buộc chặt chẽ lắm.

IV.Lưu đồ mô phỏng tiến trình của các giao thức.

1. Tiến trình giao dịch SMTP



Tiến trình giao dịch SMTP

- Tiến trình giao dịch SMTP là khá phức tạp. Nó phải thực hiện chức năng như một bộ dịch và xử lý lệnh theo chuẩn giao thức SMTP. Tất cả các lệnh gửi tới đều phải trải qua quá trình phân tích để xác định chính xác yêu cầu của người sử dụng. Hệ thống sử dụng một từ điển các lệnh của SMTP để làm công việc này.

- Sau khi lệnh đã được xác định là hợp lệ, nó sẽ được thực hiện tuỳ theo yêu cầu của người sử dụng.

- Trong toàn bộ phiên giao dịch SMTP, hệ thống sử dụng chủ yếu năm lệnh là: HELO, MAIL FROM, RCPT TO, DATA và QUIT. Các lệnh này luôn phải thực hiện theo một trình tự quy định như sau:

⇒ HELO

⇒ MAIL FROM

⇒ RCPT TO (lệnh này có thể được lập lại nhiều lần - sử dụng trong trường hợp cho nhiều người nhận).

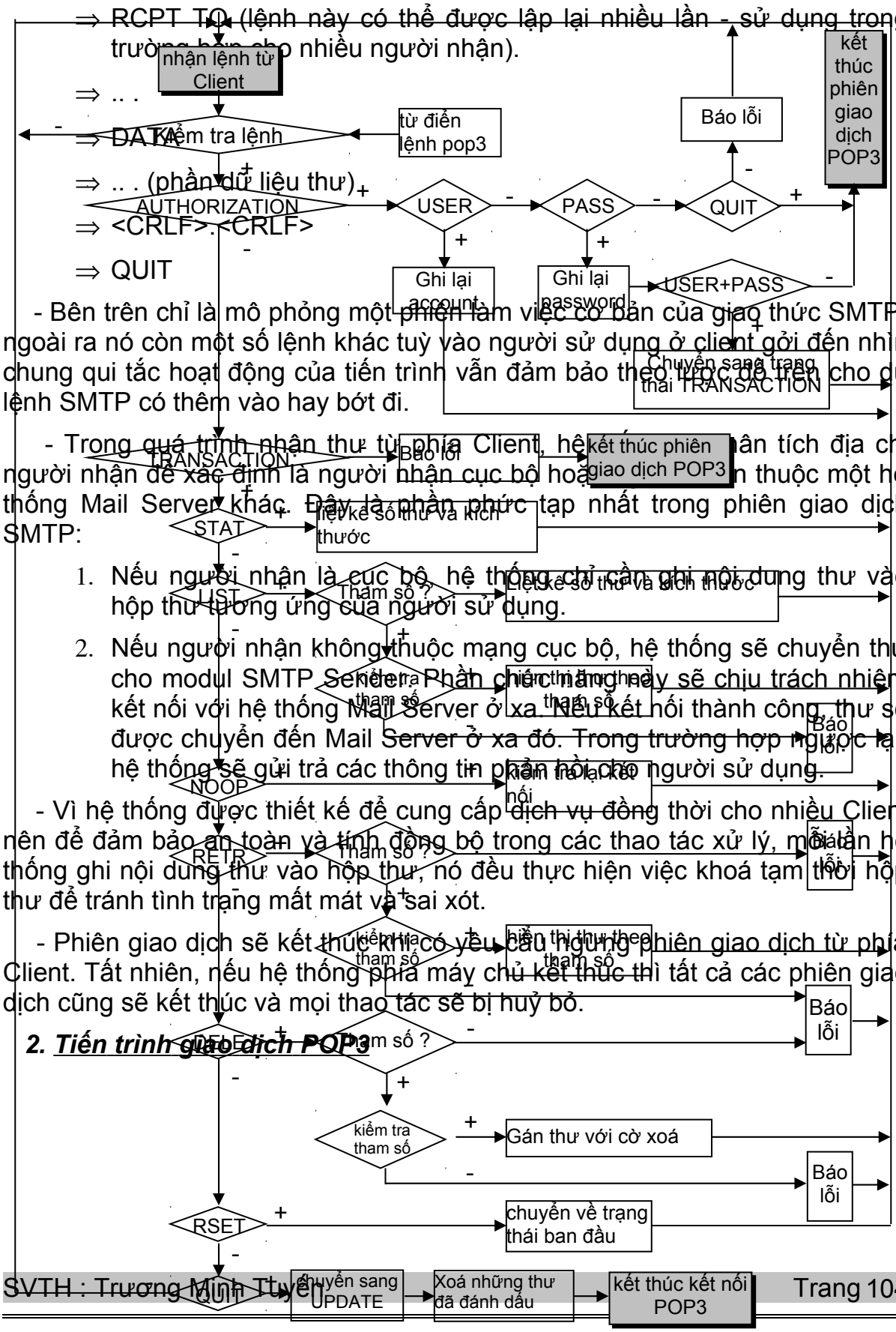
⇒ .. .

⇒ DATA

⇒ .. . (phần dữ liệu thư)

⇒ <CRLF>.<CRLF>

⇒ QUIT



TIẾN TRÌNH GIAO DỊCH POP3

- Bên trên chỉ là mô phỏng một phiên làm việc cơ bản của giao thức SMTP, ngoài ra nó còn một số lệnh khác tuỳ vào người sử dụng ở client gởi đến nhìn chung qui tắc hoạt động của tiến trình vẫn đảm bảo theo lược đồ trên cho dù lệnh SMTP có thêm vào hay bớt đi.

- Trong quá trình nhận thư từ phía Client, hệ thống phân tích địa chỉ người nhận để xác định là người nhận cục bộ hoặc là thuộc một hệ thống Mail Server khác. Đây là phần phức tạp nhất trong phiên giao dịch SMTP:

1. Nếu người nhận là cục bộ, hệ thống chỉ cần ghi nội dung thư vào hộp thư tương ứng của người sử dụng.
2. Nếu người nhận không thuộc mạng cục bộ, hệ thống sẽ chuyển thư cho modul SMTP Sender. Phần chương này sẽ chịu trách nhiệm kết nối với hệ thống Mail Server ở xa. Nếu kết nối thành công, thư sẽ được chuyển đến Mail Server ở xa đó. Trong trường hợp ngược lại, hệ thống sẽ gửi trả các thông tin phản hồi cho người sử dụng.

- Vì hệ thống được thiết kế để cung cấp dịch vụ đồng thời cho nhiều Client nên để đảm bảo an toàn và tính đồng bộ trong các thao tác xử lý, mỗi lần hệ thống ghi nội dung thư vào hộp thư, nó đều thực hiện việc khoá tạm thời hộp thư để tránh tình trạng mất mát và sai xót.

- Phiên giao dịch sẽ kết thúc khi có yêu cầu ngưng phiên giao dịch từ phía Client. Tất nhiên, nếu hệ thống phía máy chủ kết thúc thì tất cả các phiên giao dịch cũng sẽ kết thúc và mọi thao tác sẽ bị huỷ bỏ.

2. Tiến trình giao dịch POP3

- Cũng giống như tiến trình giao dịch SMTP, tiến trình giao dịch POP3 cũng khá phức tạp. Nó phải thực hiện chức năng như một bộ dịch và xử lý lệnh theo chuẩn giao thức POP3. Tất cả các lệnh gửi tới đều phải trải qua quá trình phân tích để xác định chính xác yêu cầu của người sử dụng. Hệ thống sử dụng một từ điển các lệnh của POP3 để làm công việc này.

- Sau khi lệnh đã được xác định là hợp lệ, nó sẽ được thực hiện tuỳ theo yêu cầu của người sử dụng.

- Trong toàn bộ phiên giao dịch POP3, hệ thống sử dụng chủ yếu là các lệnh: USER, PASS, STAT, LIST, RETR, DELE và QUIT. Các lệnh còn lại có thể sử dụng hoặc không. Thứ tự của các lệnh là:

⇒ USER

⇒ PASS

⇒ STAT, LIST, RETR, DELE, NOOP,..., (các lệnh này không cần theo thứ tự)

⇒ QUIT

- Mỗi lần bắt đầu phiên giao dịch, hệ thống sẽ kiểm tra các tham số người sử dụng đưa vào trong hai lệnh USER và PASS để xác định người nhận thư. Nếu việc định danh thành công thì hộp thư sẽ được mở cho người sử dụng này. Đồng thời, hệ thống sẽ cấm không cho bất kỳ ai thay đổi thậm chí sử dụng hộp thư đã bị khoá. Trong trường hợp hộp thư đã khoá, nếu có một người sử dụng nào khác cũng định mở hộp thư sẽ bị hệ thống từ chối và kết thúc luôn phiên giao dịch.

- Trong quá trình khoá hộp thư, hệ thống vẫn cho phép hộp thư của người sử dụng này nhận thư gửi đến từ phía các Client khác. Tuy nhiên, các thư mới sẽ không được sử dụng ngay trong phiên giao dịch hiện thời. Chỉ sau khi phiên giao dịch kết thúc thì toàn bộ thư mới được cập nhật vào hộp của người sử dụng.

- Trong phiên giao dịch có thao tác xoá thư. Tuy nhiên chức năng này chỉ thực hiện việc đánh dấu xoá tạm thời. Trong trường hợp người sử dụng muốn khôi phục lại thì hệ thống vẫn cho phép bằng cách sử dụng lệnh RSET và đồng thời hệ tiến trình chuyển về trạng thái ban đầu (trước khi vào trạng thái AUTHORIZATION). Trong trường hợp ngược lại, khi hệ thống đã chuyển sang trạng thái UPDATE thì mọi thư đã đánh dấu sẽ bị xoá hẳn.

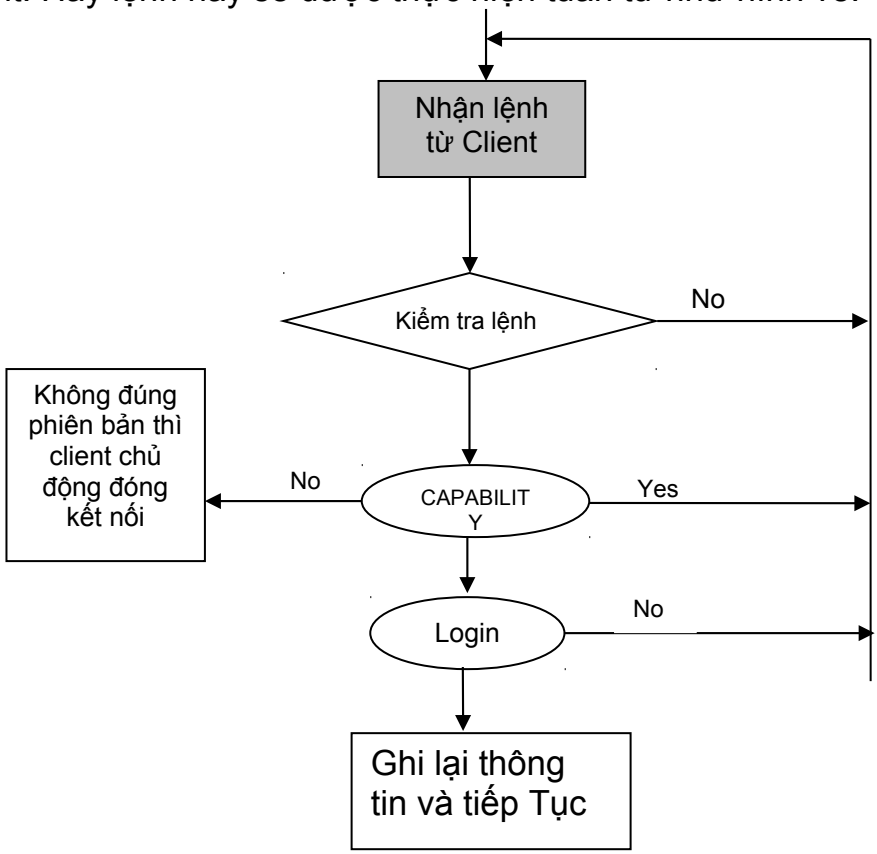
- Phiên giao dịch sẽ kết thúc khi có yêu cầu ngừng phiên giao dịch từ phía Client. Hoặc trong trường hợp có lỗi như đã nêu trên. Tất nhiên, nếu hệ thống phía máy chủ kết thúc thì tất cả các phiên giao dịch cũng sẽ kết thúc và mọi thao tác sẽ bị huỷ bỏ.

3.Tiến trình giao dịch imap4

- Tiến trình của IMAP4 phức tạp hơn nhiều so với POP3 vì thư viện lệnh nhiều hơn và tất cả điều được xử lý trên Server, dưới đây là một số tiến trình cơ bản mà khi kết nối với IMAP4 mà mọi người dùng điều phải đi qua.

- Trước tiên, kiểm tra xem phiên bản imap đang dùng trên Server có phải là phiên bản 4.0 không, nếu đúng thì tiếp tục tiến trình kiểm tra lệnh login khi

login thành công thì tuỳ theo những lệnh tiếp theo mà client yêu cầu còn không kể như tiến trình đã kết thúc mà sự chủ động kết thúc tiến trình này là từ Client. Hay lệnh này sẽ được thực hiện tuần từ như hình vẽ.



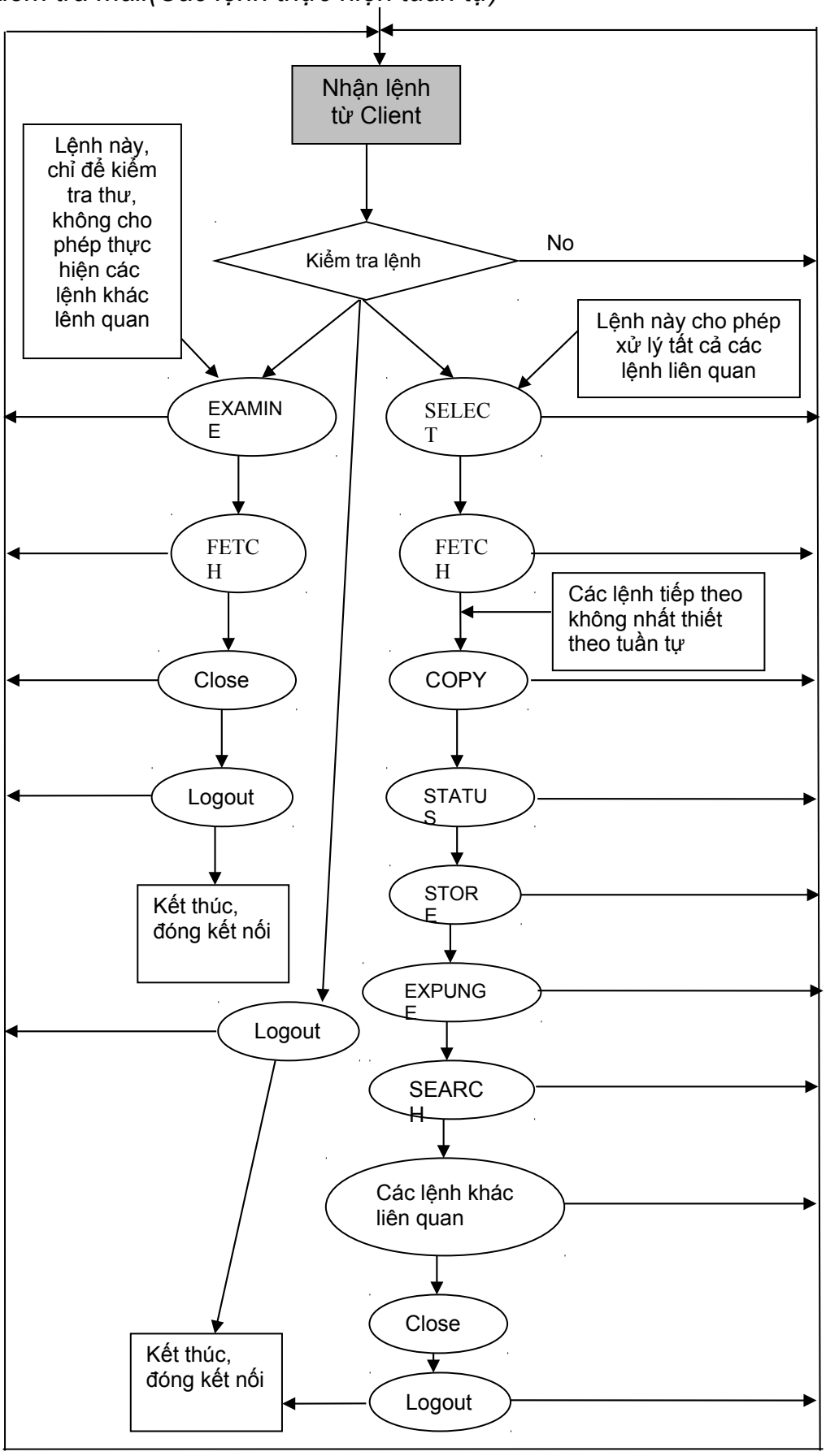
- Lệnh IMAP cụ thể như sau

C: 0001 CAPABILITY
S: * CAPABILITY IMAP4rev1 // IMAP4rev1 là phiên bản 4.0
S:0001 OK CAPABILITY completed
C:0002 LOGIN tuyentm01 kimphung
S: 0002 OK LOGIN completed

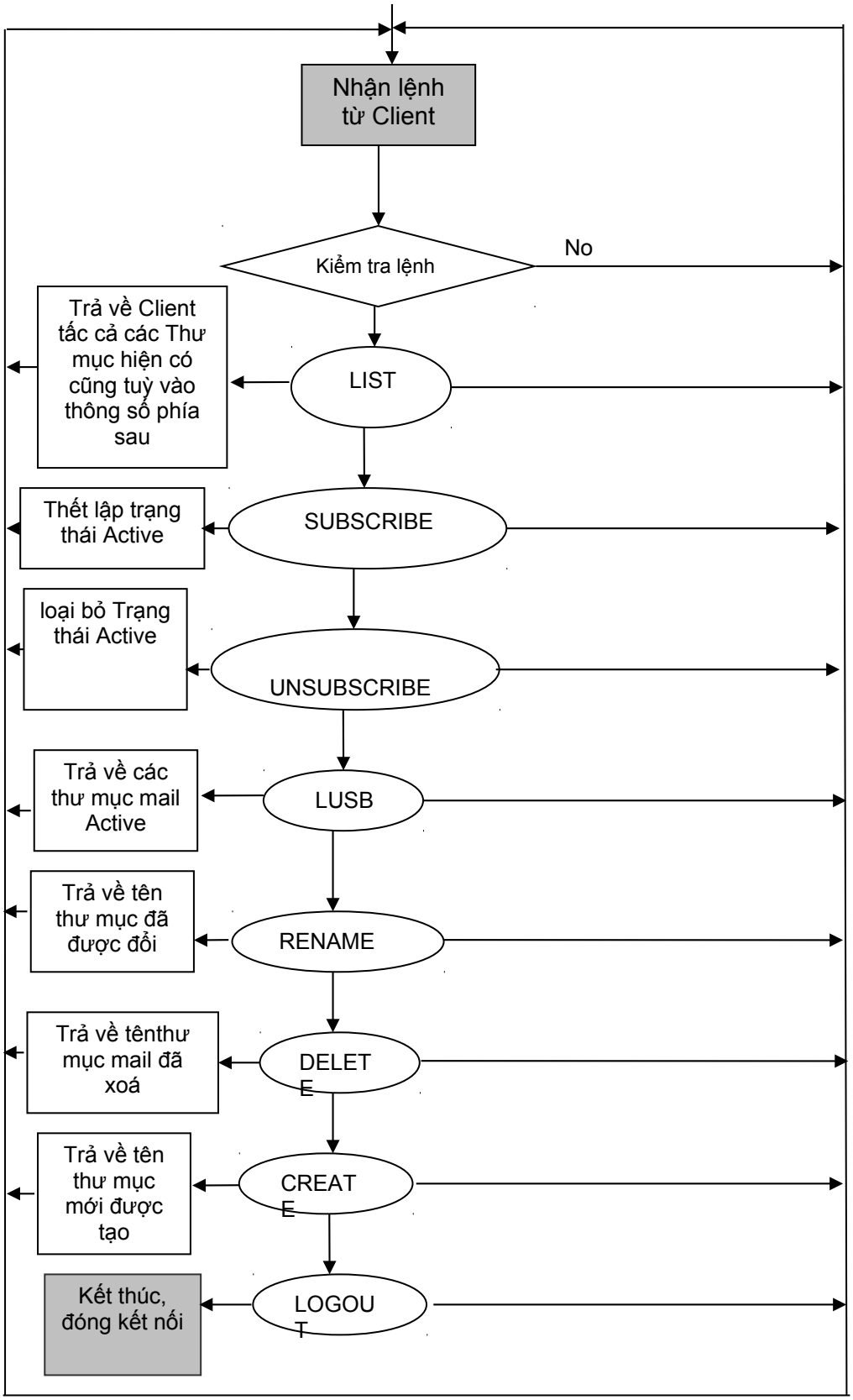
Sau khi kiểm tra phiên bản Imap và login thành công tiếp theo tuỳ từng người sử dụng mà có những phiên làm việc tiếp theo như là kiểm tra mail và thiết lập lại trạng thái mail hay kiểm tra thư mục(hòm thư) bao gồm tạo hòm thư mới, đổi tên hay xoá một hòm thư. Nhìn chung nó có 2 hướng cụ thể là kiểm tra mail từng hòm thư hay kiểm tra hòm thư và xử lý.

Sau khi login thành công thì tiến trình tiếp tục như sau

a. kiểm tra mail(Các lệnh thực hiện tuần tự)



b.tiến trình xử lý hộp thư (Các lệnh không nhất thiết tuần tự)

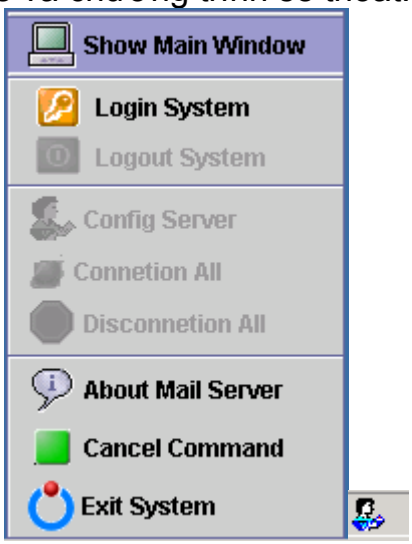


- Những mô hình trên chỉ là mô hình miêu tả một dạng xử lý tiến trình cơ bản của IMAP, tuỳ vào từng chương trình mail client mà những mô hình trên có thể thêm lệnh hay loại bỏ bớt lệnh. Các lệnh trong giao thức Imap có phụ

thuộc lẫn nhau như để truy xuất mail từ thư mục mail thì trước tiên là phải gọi lệnh select hay examine trước rồi mới tới các lệnh tiếp theo mới có hiệu lực, ….

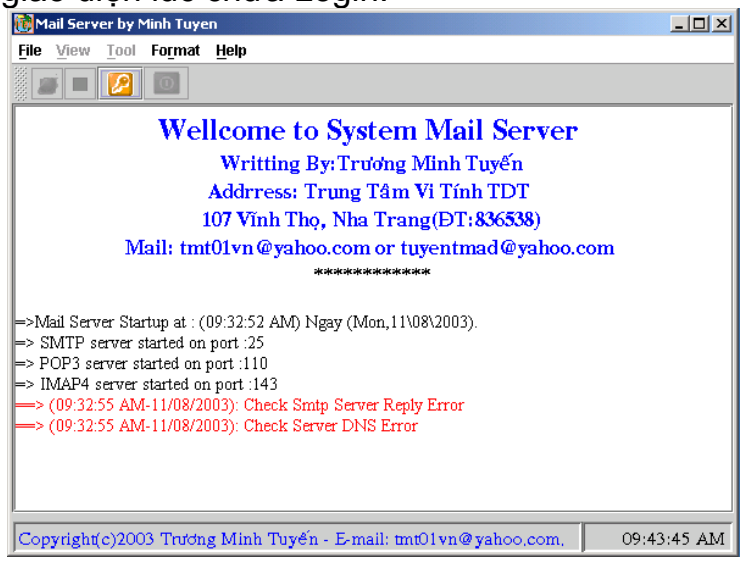
V.Một số hình ảnh minh hoạ và diễn giải chương trình.

- Sau khi khởi đông xong chương trình trên server, nếu không gặp lỗi thì chương trình được nạp xuống thanh Systray nếu hệ điều hành đang dùng là họ Windows. Còn gặp lỗi như cơ sở dữ liệu chưa được khởi tạo trong ODBC hay cổng POP3, SMTP, IMAP4 đã được sử dụng lúc đó chương trình sẽ hiện ra hộp thông báo lõi cụ thể và chương trình sẽ thoát.

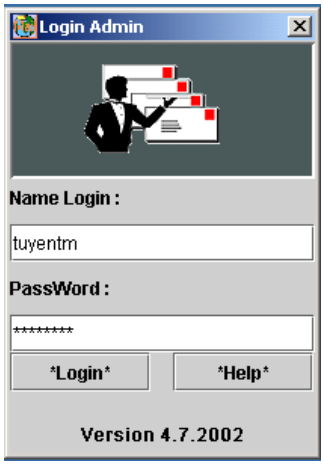


- Khi chương trình đang chạy nếu chưa login vào hệ thống thì chương trình đang ở trạng thái treo, có nghĩa là không tắc được chương trình và cũng không chỉnh đổi thay những thao tác khác trên đó nhưng chương trình vẫn lắng nghe kết nối từ Client.

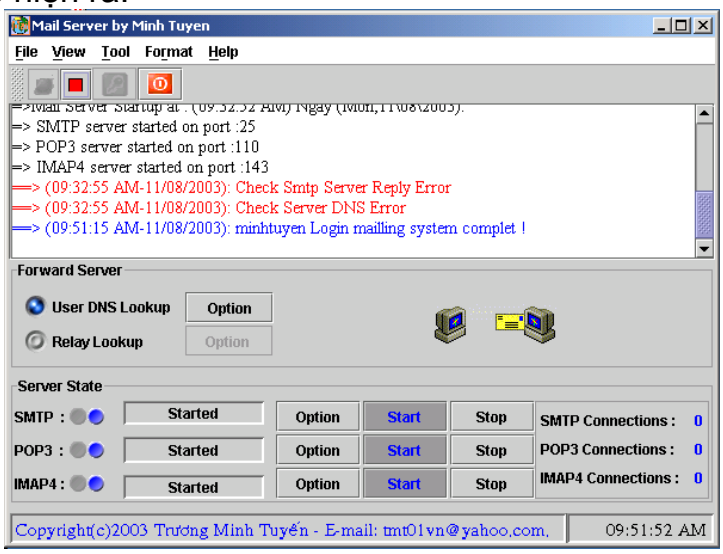
+ Màn hình giao diện lúc chưa Login.



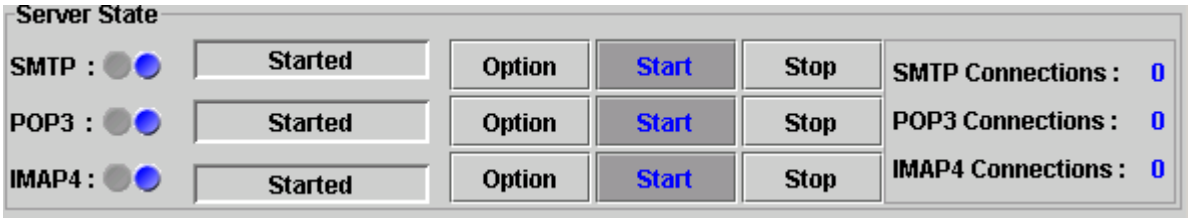
+ Giao diện login vào chương trình



+ màn hình giao diện lúc Login thành công lúc này mọi điều khiển trong chương trình sẽ hiện ra.



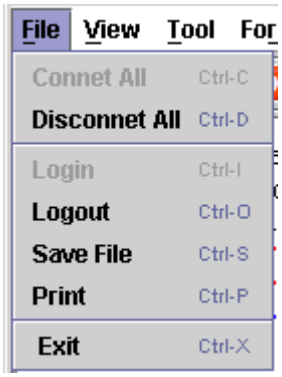
- Ngoài giao diện chính khi login sẽ cho biết trạng thái Server của các giao thức ta có thể cho ngừng lắng nghe từ Client hay chỉnh đổi cấu hình các Server trên.



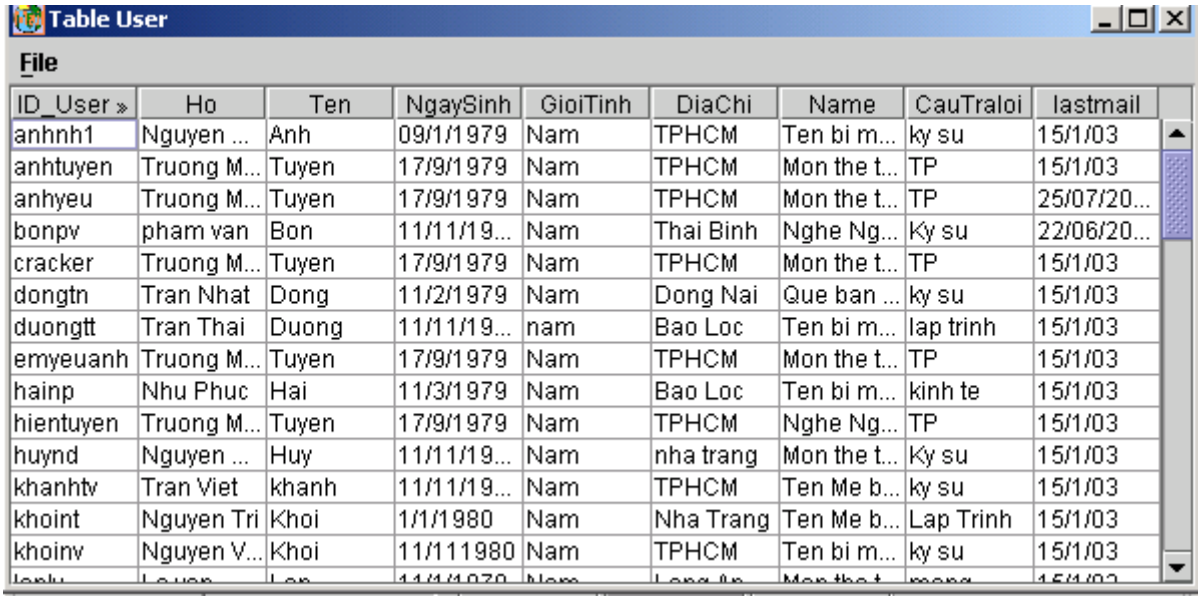
- Khi nhận mail mà không thuộc quyền quản lý của Server thì hộp thoại sau sẽ làm nhiệm vụ như trạm trung gian chuyển mail đến đích.

như hình trên thì ServerForward được chọn làm nhiệm vụ này, ta có thể sửa đổi lại cấu hình của Server này.

- Khi login xong trên thanh toolbar có 4 biểu tượng bao gồm connect và Disconnect, Login và logout, có chức năng tương ứng từng tên gọi. Nhưng biểu tượng này có thể tìm thấy trong menu File trong menu này còn có chức năng Save File và Print hay chức năng này điều khiển khung chữ nhìn thấy trên giao diện chính.



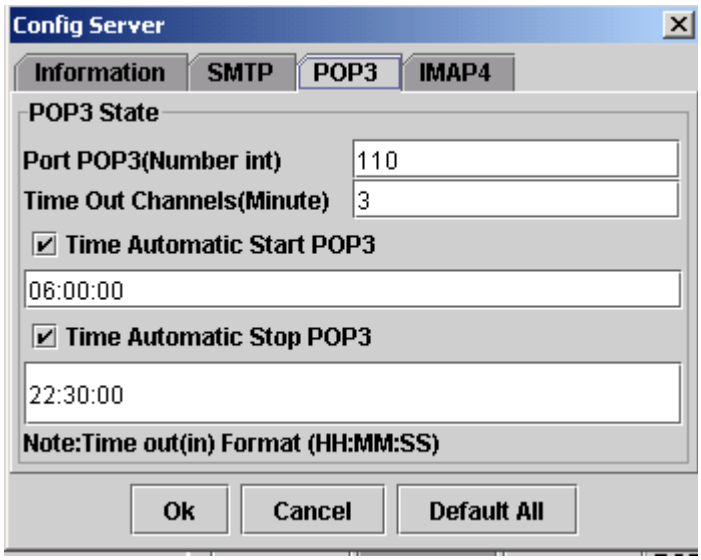
- Menu View, là menu xem xem thông tin và các panel điều khiển. Như thể hiện tất cả thông tin người dùng hiện chương trình đang quản lý.



ID_User	Ho	Ten	NgaySinh	GioiTinh	DiaChi	Name	CauTraloi	lastmail
anhnh1	Nguyen ...	Anh	09/1/1979	Nam	TPHCM	Ten bi m...	ky su	15/1/03
anhtuyen	Truong M...	Tuyen	17/9/1979	Nam	TPHCM	Mon the t...	TP	15/1/03
anhyeu	Truong M...	Tuyen	17/9/1979	Nam	TPHCM	Mon the t...	TP	25/07/20...
bonpv	pham van	Bon	11/11/19...	Nam	Thai Binh	Nghe Ng...	Ky su	22/06/20...
cracker	Truong M...	Tuyen	17/9/1979	Nam	TPHCM	Mon the t...	TP	15/1/03
dongtn	Tran Nhat	Dong	11/2/1979	Nam	Dong Nai	Que ban ...	ky su	15/1/03
duongtt	Tran Thai	Duong	11/11/19...	nam	Bao Loc	Ten bi m...	lap trinh	15/1/03
emyeuanh	Truong M...	Tuyen	17/9/1979	Nam	TPHCM	Mon the t...	TP	15/1/03
hainp	Nhu Phuc	Hai	11/3/1979	Nam	Bao Loc	Ten bi m...	kinh te	15/1/03
hientuyen	Truong M...	Tuyen	17/9/1979	Nam	TPHCM	Nghe Ng...	TP	15/1/03
huynd	Nguyen ...	Huy	11/11/19...	Nam	nha trang	Mon the t...	Ky su	15/1/03
khanhtv	Tran Viet	khanh	11/11/19...	Nam	TPHCM	Ten Me b...	ky su	15/1/03
khoint	Nguyen Tri	Khoi	1/1/1980	Nam	Nha Trang	Ten Me b...	Lap Trinh	15/1/03
khoinv	Nguyen V...	Khoi	11/111980	Nam	TPHCM	Ten bi m...	ky su	15/1/03

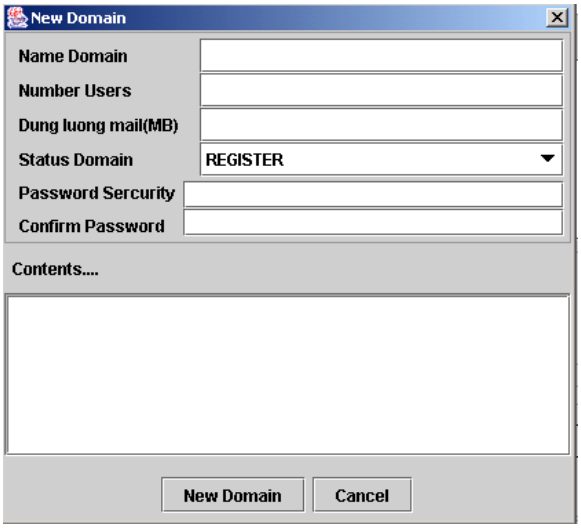
- Menu Tool

+ Đây là menu điều khiển chính của chương trình như thiết lập lại ngày giờ tự động cho các giao thức SMTP Server, POP3 Server, IMAP4 Server.



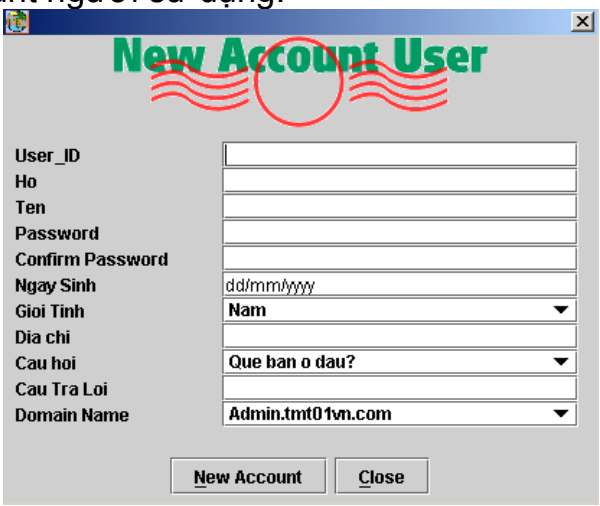
Theo hình trên ta đang thiết lập giờ cho giao thức Pop3 như cổng Client kết nối vào là 110, giờ tự động kết nối đang được chọn tại thời điểm 06:00:00 và giờ đóng kết nối là 22:00:00.

+ Tạo Tên miền



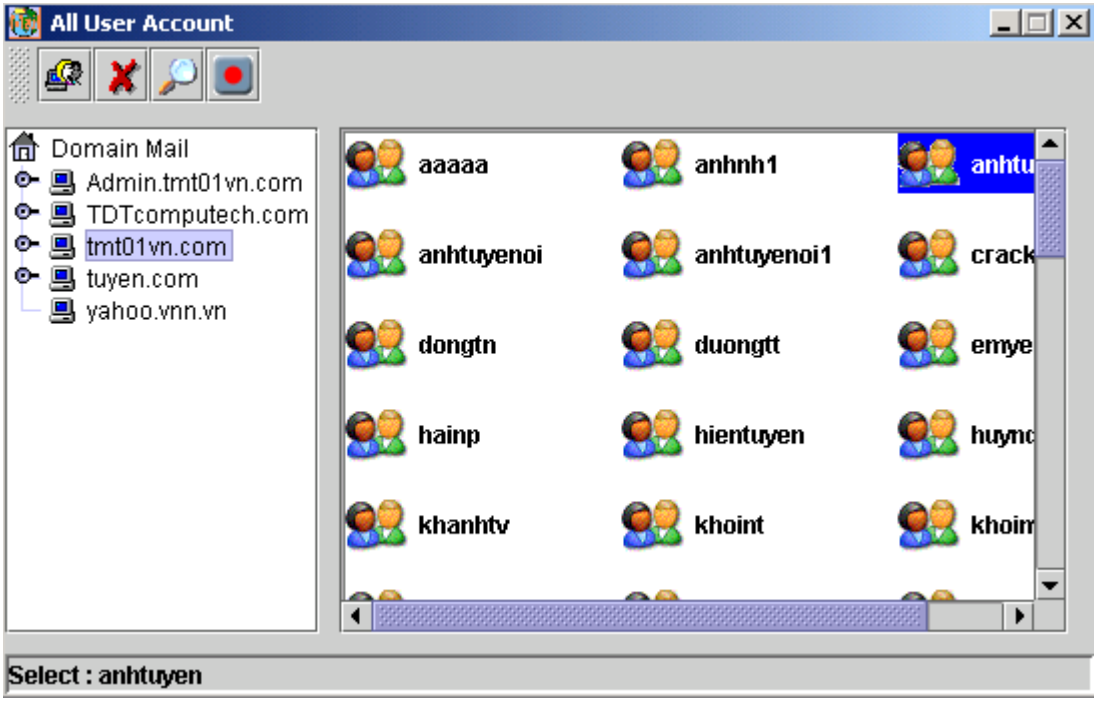
Tên miền được tạo ra có hay loại đó là miễn phí hay tên miền đã được đăng ký, tất cả được thể hiện trên dialog đăng lý tên miền.

+ Tạo một Account người sử dụng:



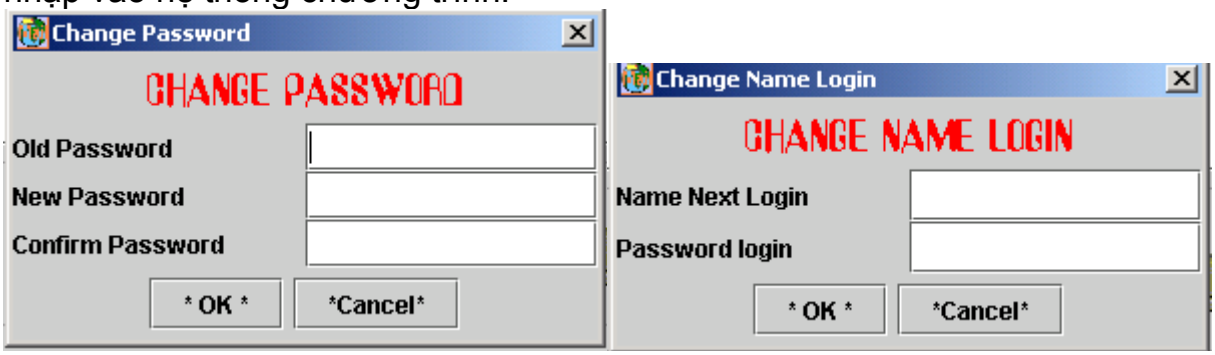
Đăng ký một thành viên mới thuộc bất kỳ một Tên miền nào dựa vào bản đăng lý trên sau khi đăng ký thì tên miền này được tồn tại và thuộc quyền quản lý của tên miền được tạo ra.

- Ngoài ra để cập nhật thông tin cũng như tên miền thì hộp dialog từ menu tool sau sẽ làm việc này. Ở đây ta thấy được thành viên nào thuộc tên miền nào và ta cũng có thể tạo thêm tên miền, tạo Account mới, xoá Account hay khoá tên miền lại (tạm thời không sử dụng) và cũng có thể xoá hẳn một tên miền nào đó.



Bên trái là tên miền và bên phải là những Account thuộc tên miền đó,ở hộp dialog này còn rất nhiều tính năng khác nó tương tự như một cây thư mục trong Windons.

- Cũng từ menu tool này ta có thể thay đổi mật khẩu cũng như tên đăng nhập vào hệ thống chương trình.



- Menu Format : chỉ là một tiễn ích làm cho chương trình thêm sinh động như chỉnh lại Font, màu chữ, màu nền,…

- Menu Help : chỉ là menu trợ giúp người sử dụng.

- Ngoài những tính năng chủ yếu trên chương trình còn rất nhiều tính năng khác khá hay khi chạy chương trình sẽ thấy rõ được điều đó. Nó tương đối thân thiện với người dùng và dễ sử dụng.

VI.Cài đặt và chạy chương trình.

- Vì chương trình được viết bằng ngôn ngữ Java mã nguồn mỡ, nên để chạy được chương trình thì điều kiện cần trước tiên là hệ điều hành dùng để chạy chương trình phải hiểu Java nghĩa là trên máy phải có máy ảo Java đang chạy. Để có được máy ảo Java tốt nhất nên cài đặt JDK mọi phiên bản của hãng Sun,khi cài đặt xong JDK chỉ cần chạy File Server.bat thì chương trình trên Server sẽ được thực thi.
- Cần phải cài đặt cơ sở dữ liệu trong cầu nối ODBC với tên là AccountMail và chọn đường dẫn đến đúng file cơ sở dữ liệu trên đĩa.

VII.Hướng phát triển chương trình.

- Nhìn chung chương trình được thực hiện khá hoàn hảo cho một dịch vụ mail Server hỗ trợ đa miền, có tính bảo mật chặt chẽ, dễ hiểu dễ sử dụng. Nhưng nó vẫn còn những mặt cần được phát triển lên.

+ Chương trình mới chỉ ở mức thử nghiệm, chưa thực sự hoạt động trên Internet và đây cũng là một hướng mà chương trình cần được phát triển hỗ trợ với dịch vụ Web Server tạo nên một dịch vụ khép kín.

+ Mỗi một tên miền chưa có người quản trị cụ thể đây cũng là mặt hạn chế và được phát triển sau.

+ Chương trình chưa hỗ trợ dịch vụ đăng ký mail list và gởi mail list định kỳ, đây cũng là một khả năng rất hay của dịch vụ mail Server.

CHƯƠNG 6

PHÂN TÍCH, XÂY DỰNG VÀ CÀI ĐẶT CHƯƠNG TRÌNH WEB MAIL TRÊN MÁY CLIENT

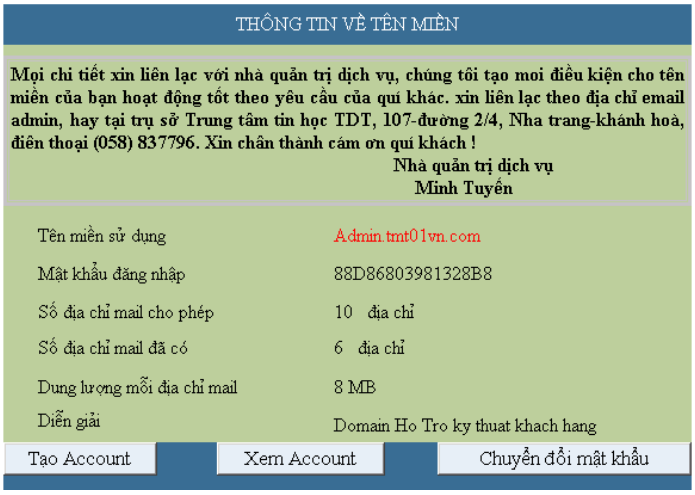
- Web mail là một dịch vụ mail client rất nổi tiếng và thông dụng nhất hiện nay trên Internet. Web mail mà em sẽ trình bày cũng dựa vào những đặc tính trên. Mỗi một Web mail chỉ truy xuất và quản lý của một mail Server nào đó, Web mail này được tạo ra nhằm mục đích thể hiện rõ hơn những tính năng mà chương trình mail Server được thiết kế ở trên.
- Web mail này dùng giao thức IMAP4 và SMTP để đọc và gởi thư. Nó cũng có những tính năng rất tiễn ích và rất thân thiện với người dùng.
- Web mail được thiết kế bằng ngôn ngữ JSP và trình điều khiển là Jrun của hãng allaire, lệnh truy xuất bằng giao thức SMTP và IMAP4 được kế thừa từ từ các lớp của hãng Sun, em không có viết lệnh SMTP Client và IMAP4 Cloient mà chỉ kế thừa rồi sử dụng.

I.Các giao diện của chương trình.

1. Giao diện đăng ký Account.

- Để đăng ký một Account miễn phí từ Web mail, như ở hình trên nhấn chuột vào nút đăng lý mail mới lúc đó trang đăng ký mail mới được mở ra và điền đầy đủ những thông tin cần thiết, nếu thành công thì Account mới được tạo ra.

- Để tạo một Account mới không phải miễn phí thì trước tiên phải đăng nhập vào tên miền mà bạn đã đăng ký với dịch vụ, rồi sau đó có những mục tự chọn như thêm, xoá Account,… Nếu như Domain bạn bị khoá lại hay số lượng người dùng đã giới hạn thì không thể nào thêm Account mới.



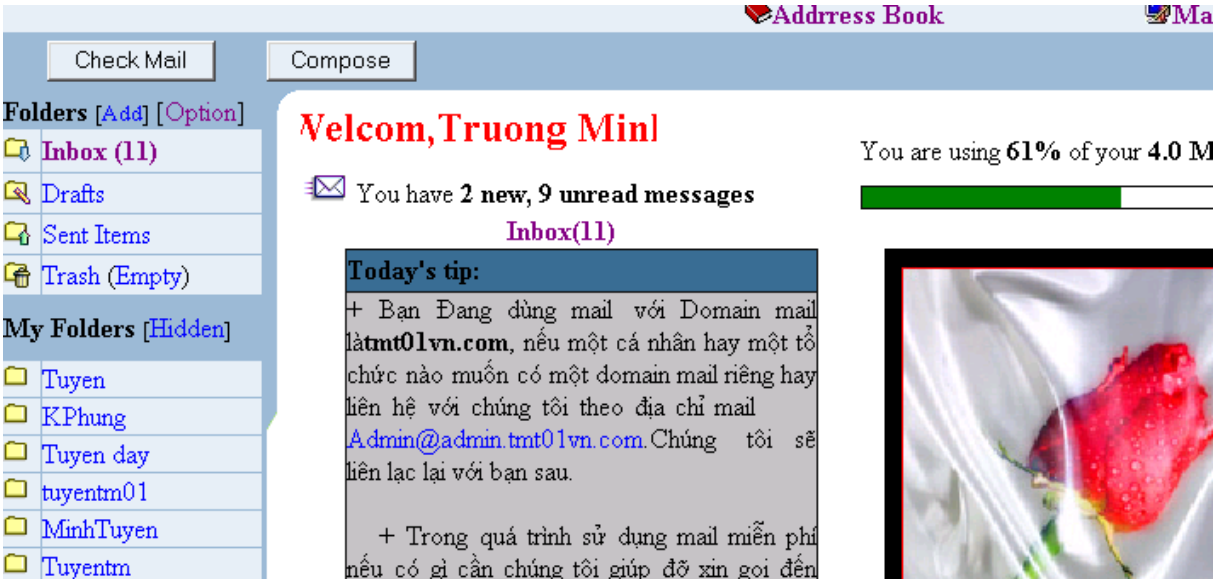
2 .Giao diện kiểm tra mail.

- Để kiểm tra mail người dùng phải đăng nhập vào hệ thống, giao diện đăng nhập như sau.

- Nếu trong quá trình sử dụng lỡ quên mật khẩu thì có thể tìm lại mật khẩu thông qua câu hỏi và câu trả lời mà người dùng đã đăng ký với Server mail. Đây là một tiễn ích khá hay mà bất kỳ một Web mail nào cũng hỗ trợ nó.

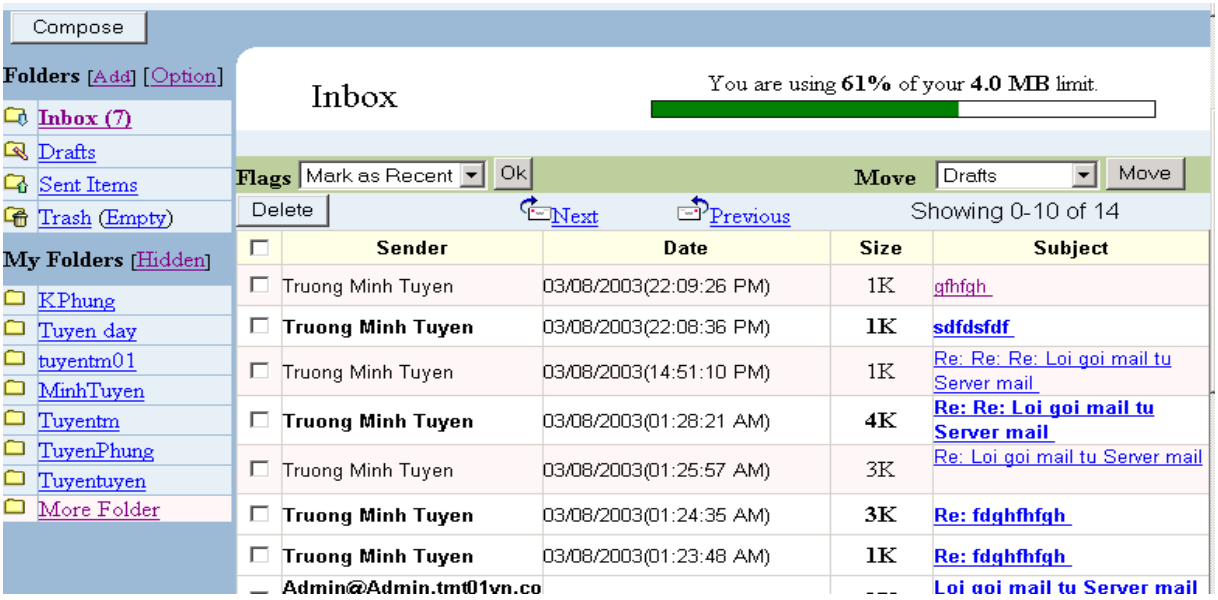
- Sau khi đăng nhập thành công thì trang kiểm tra mail được mở ra, chào mừng thông báo là có bao nhiêu thư mới trong hộp thư Inbox và thể hiện các tính năng khác mà người dùng đã cấu hình trước đó. Khi vào trang này vì giao thức sử dụng là IMAP nên ta dễ dàng biết được có bao nhiêu thư mới và bao nhiêu thư chưa đọc trong những lần thăm mail trước đó thông qua việc kiểm tra cờ trạng thái của từng lá thư. Sau đây là giao diện thể hiện thông tin

.

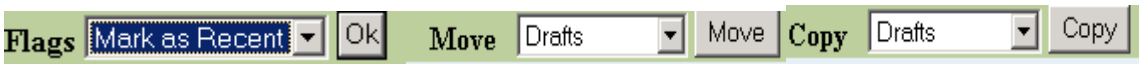


Theo hình trên thì hộp thư có 2 thư mới và 9 lá thư cũ chưa đọc, phía bên phải là những hộp thư mail mà thuộc tính của nó là Active, để vào xem thông tin từng lá thư trong hộp thư ta nhấn chuộc vào từng thư mục thư tương ứng, còn nút check mail là check hộp thư mặt định inbox. Trang này ta có thể cấu hình lại các hộp thư thì nhấn vào Optiont tại Folders và có thể thêm hay xoá bớt hộp thư. Thư sau khi được xoá ở các hộp thư khác sẽ được mail Server lưu vào hộ thư Trash(đây là hộp thư rác) và xoá thư ở hộp thư này thì thư mới loại ra khỏi Account người dùng Chỉ có giao thức IMAP mới có chức năng quản lý đa hộp thư còn POP3 thì chỉ có duy nhất một hộp thư là Inbox.

- Giao diện kiểm thể hiện thông tin từng lá thư trong hộp thư sau.

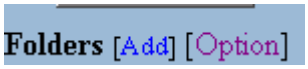


vào đây ta có thể thấy rõ người gởi, ngày gởi, kích thước và chủ đề từng lá thư, để xem nội dung từng lá thư nhấn chuột vào cột Subect từng lá thư một. Ngoài ra ta còn biết được thư nào mới và thư nào chưa đọc thông qua màu nền của từng lá thư, thư mới và chưa đọc có màu nền trắng, thư đọc rồi có màu vàng nhạt. ở đây có một điều thú vị nữa rất hay từ giao thức imap đó là chúng ta có nhiều khả năng thao tác trên thư hơn như thiết lập lại trạng thái lá thư, copy, move thư sang hộp thư khác, Thư đã khi xoá đi người dùng có thể phục hồi lại hay thiết lập lại trạng thái lá thư,…..



3. Giao diện thao tác Thư mục mail

- Để thao tác và xử lý thư mục mail như thêm mới, đổi tên, xoá hay thiết lập trạng thái Active(show) từng thư mục hãy vào mục option. Mục này sẽ làm tất cả những gì liên quan đế thư mục mail, ngoài ra ta có thể nhấn chuột vào add để thêm một thư mục mail mới mà không cần vào Option.



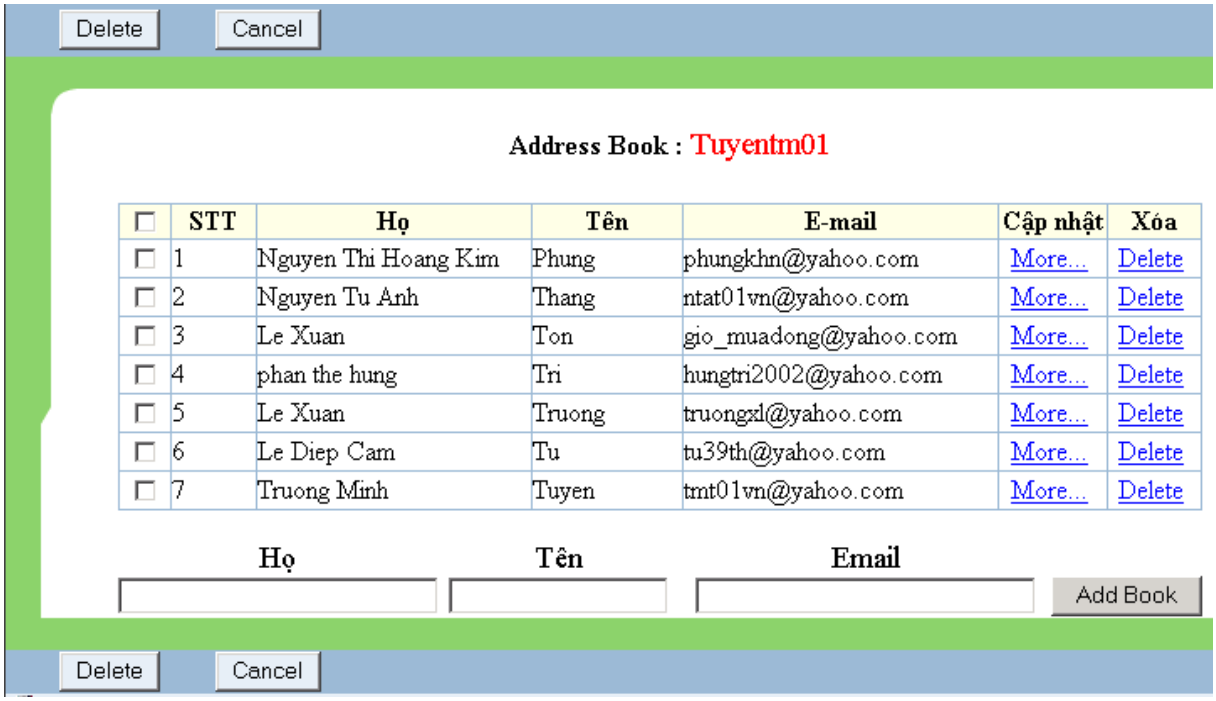
Trang option có giao diện như sau:

	Full Name Folder	Attribute	Messages	Delete	Rename	Check Mail
	Inbox	Show	14	No	No	Check
	Drafts	Show	0	No	No	Check
	Sent Items	Show	3	No	No	Check
	Trash	Show	0	No	No	Check
	Tuyen	Show	0	Delete	Rename	Check
	Tuyen/bbbbbbbb	Hidden	0	Delete	Rename	Check
	Tuyen/tuyentm	Hidden	0	Delete	Rename	Check
	Tuyen/tuyentm/inbox	Hidden	0	Delete	Rename	Check
	KPhung	Show	1	Delete	Rename	Check
	Tuyen day	Show	0	Delete	Rename	Check
	tuyentm01	Show	0	Delete	Rename	Check
	MinhTuyen	Show	2	Delete	Rename	Check
	MinhTuyen/tuyen	Hidden	0	Delete	Rename	Check
	Tuyentm	Show	0	Delete	Rename	Check

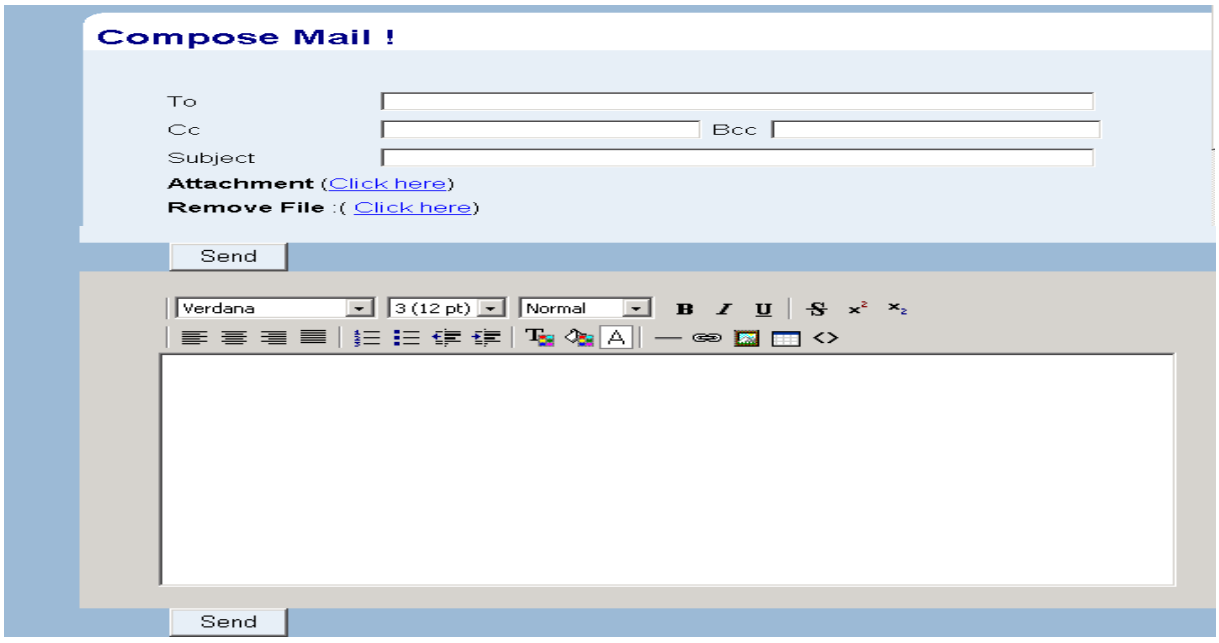
khi vào đây ta biết được hộp thư đó đang ở trạng thái nào có bao nhiêu lá thư, nếu hộp thư đang ở trạng thái Hidden thì hộp thư này không thấy trong giao diện sau khi login.

4.Một số giao diện khác.

- Web mail còn hỗ trợ sổ địa chỉ nghĩa là người sử dụng lưu lại địa chỉ mail của bạn bè người thân,… Sổ địa chỉ này lưu trữ họ, tên, ngày sinh, email,…. Giao diện sổ địa chỉ như sau.



- Giao diện soạn thư, vì giao diện có dùng java Script nên để thể hiện hết các tính năng của một trình soạn thảo nên chạy trên môi trường Explorer 5.5 trở lên. Giao diện hỗ trợ đầy đủ các tính năng cơ bản mà người dùng cần đến. Người dùng có thể gởi mail với một file đính kèm theo, chương trình chỉ mới xử lý được kèm một file đây cũng mà một mặt còn hạn chế của Web mail mà cần được cải thiện. Ngoài hai chức năng mail CC, BCC, người dùng có thể gởi mail đến nhiều người khác bằng cách mỗi địa chỉ mail cách nhau dấu chấm phẩy ";".



- Ngoài ra người sử dụng có thể sửa đổi thông tin đăng ký ban đầu khi nhấn chuột vào mail Option.

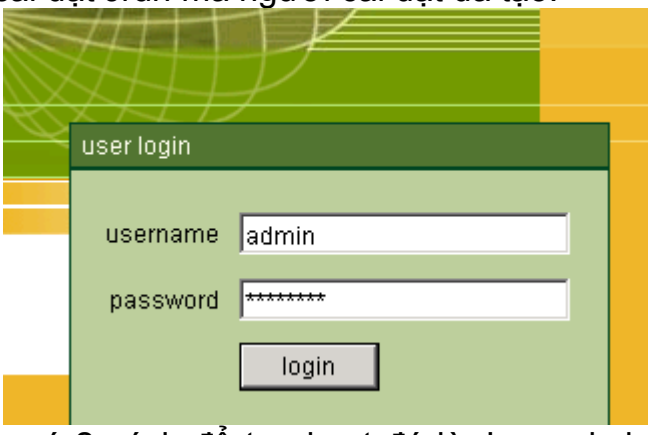
- Ngoài những tính năng trên Web mail còn một số tính năng khác nhìn chung rất thân thiện và dễ sử dụng.

II. Cách cài đặt và xử lý chương trình Web mail.

- Máy chủ để lưu trữ trang Web mail này phải cài đặt chương trình Java Server page đó là Jrun.

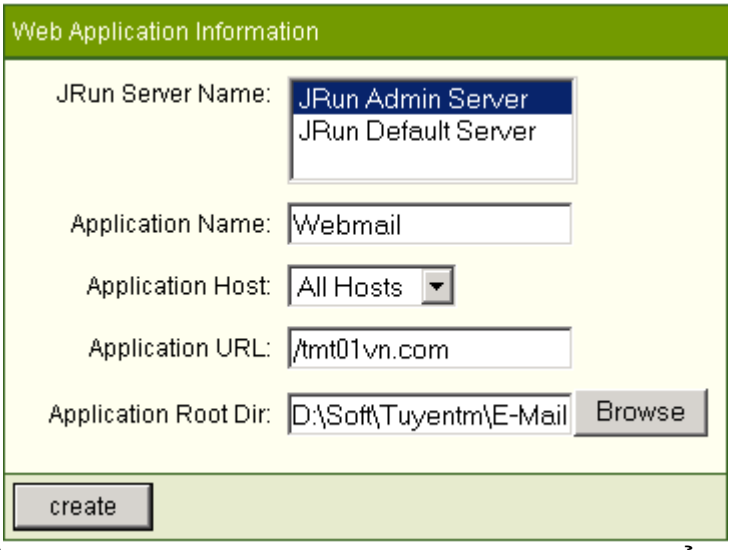
-Tiếp theo phải tạo một host để cho máy Client hiểu và chạy được. Quy trình tạo host băng Jrun như sau.

+Trước tiên login vào Jrun tên đăng nhập mặt định là admin còn password là trong quá trình cài đặt Jrun mà người cài đặt đã tạo.



+ Khi login vào có 2 cách để tạo host đó là Jrun admin Server trên cổng 8000 hoặc Jrun Default Server trên cổng 8100.

theo hình thì host được tạo ra từ cổng 8000(Admin Server) chọn mục Web Applications, trang này được mở ra nhấn chuột vào create an Applications.



Sau khi điền đầy đủ những thông tin như tên ứng dụng, cổng ứng dụng, địa chỉ URL ứng dụng cuối cùng là thư múc link đến ứng dụng trên đĩa. Như hình trên sau khi tạo xong thì host ứng dụng có tên : http://tênmáychủ:8000/tmt01vn.com/.

+ tiếp theo từ trình duyệt Explorer máy Client đánh

http://tmt:8000/tmt01vn.com/LoginMail.jsp

đó chính là địa chỉ URL của Web mail đã đăng ký với Jrun, sau khi đăng ký xong còn một việc cần làm trên máy Server nữa là copy 2 gói tin mail.jar và activation.jar vào thư mục của Jrun đó là root_Jrun/Allaire/Jrun/servers/default/default-app/WEB-INF/lib/.

III. Những mặt hạn chế của Web mail.

- Web mail cũng có những mặt hạn chế sau như gởi mail chỉ mới gởi được một file đi kèm.
- Chương trình mới chạy trên mạng cục bộ, chưa có host riêng để thực hiện trên mạng Internet.
- Web mail chưa tạo và cài đặt book lock các địa chỉ mail gởi đến mà không thích nhận.
- Web mail chỉ check mail với giao thức IMAP chưa có mục chọn check mail với POP3.
- Chưa tạo danh sách và đăng ký mail list.